Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 1311–1316

www.elsevier.com/locate/procedia

55th CIRP Conference on Manufacturing Systems

Dynamic Partial Reconfiguration for Adaptive Sensor Integration in Highly Flexible Manufacturing Systems

Florian Schade[a,*], Christian Karle[a], Edgar Mühlbeier[b], Philipp Gönnheimer[b], Jürgen Fleischer[b], Jürgen Becker[a]

[a]*Institute for Information Processing Technologies (ITIV), Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*
[b] *wbk Institute of Production Science, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Germany*

Abstract

As new production system concepts emerge to face an increasing demand for individualized production, sensor integration for Industry 4.0 functions is becoming a major challenge. Flexibility- and scalability-focused concepts rely on reconfigurable machines that automatically replace smart end effectors between production steps. To ensure adaptability to future demands, these smart components, comprising the main tool and sensors, are connected to the static part of the machine via a unified electro-mechanical interface. In robot-based concepts, these components are furthermore space- and weight-constrained and should be cost-optimized. With respect to sensor integration, this leads to the challenge of interfacing sensors using different communication protocols and electrical signal properties through a fixed, minimal set of signal lines. In this paper, an architecture for an adaptive sensor integration unit is presented, targeting highly flexible production systems. Leveraging dynamic partial FPGA reconfiguration to exchange communication logic and an extensible hardware module located in the static part of the machine, it efficiently supports alternating communication protocols over static signal lines. It thereby reduces the number of signal lines as well as the need for protocol conversion units in the exchangeable components. A prototype implementation using industrial bus protocols shows its suitability and is evaluated concerning relevant timing characteristics and resource usage.

Keywords: Reconfiguration; Sensor; Flexible manufacturing system (FMS)

1. Introduction

With a continuously increasing demand for mass customization and individualized products as well as decreasing product life cycle periods, production technologies and production systems of the future face new challenges. Since the development, construction and commissioning of production plants are usually long-term scheduled and cost-intensive processes, rigid production systems are becoming increasingly unattractive in a rapidly changing market. In the future, production systems must be designed from the very beginning for different product variants or even different product families. The concept of *Wertstromkinematik* (WSK, Value Stream Kinematics) [1] addresses this need. It tackles this issue by proposing a highly flexible, robot-based production concept eliminating the need for special machines and instead implementing all production processes using unified, flexible, robot-like kinematics. To realize different processing steps, these robot kinematics use exchangeable robot end effectors, comprising tools and sensors as required for production, monitoring, and quality control steps. Between production steps, robots automatically change their end effector, loading the correct tool and sensors for the next process. However, implementation of complex manufacturing processes requires more powerful and closer cooperating robot kinematics, exceeding the capabilities of today's industrial robots. In WSK, this is tackled by temporarily coupling multiple robot kinematics to form a parallel kinematic structure in order to increase stiffness and achieve higher machining accuracy with higher process forces.

The WSK concept enables novel and more diverse process sequences, but at the same time requires more frequent changes of robot end effectors. In addition, the number of end effec-

* Corresponding author. Tel.: +49-721-608-41972. *E-mail address:* florian.schade@kit.edu

tors to be used by a robot increases, compared to traditional robot cells, especially when considering that tools, and thereby end effectors, shall be usable by all kinematics and the integration of future components shall be possible. Therefore, a static, unified interface is needed between robots and end effectors in the production system. When changing smart end effectors, not only mechanical connections but also electrical and communication connections, among others, must be separated or linked. For this purpose, so-called feed-through modules are used in autonomous end effector changing systems. Such modules are available for power supply, pneumatics, hydraulics and control signals as well as bus communication. To cover the whole spectrum of sensors typically used in production technology, such as cameras, temperature sensors, proximity sensors, ultrasonic sensors, encoders or acceleration and force sensors, a large variety of communication protocols must be supported. However, the available space for attaching feed-through modules is limited and expensive, hampering the introduction of more and more sensor signal lines. Also, the additional weight of the modules cannot be neglected. Therefore, to achieve the flexibility envisioned in the WSK concept while keeping weight and costs low, it must be possible to transmit many different protocols on a limited number of physical wires.

This work presents a concept to tackle this challenge. A sensor integration unit architecture is proposed, which allows for interfacing sensors using multiple communication protocols over a minimal, static set of communication lines. To achieve this in an efficient way, the dynamic partial reconfiguration (DPR) feature of field-programmable gate arrays (FPGAs) is exploited to implement alternating communication logic while an extensible PHY-board is used to realize the physical-layer implementations of the communication protocols. After a brief discussion of related work in Section 2, the proposed concept is described in Section 3. An implementation realizing relevant industrial communication protocols is presented and evaluated concerning performance and resource usage in Section 4. Its suitability is then discussed in Section 5.

2. Related work

Challenges concerning multi-protocol sensor integration in robot-based manufacturing systems have hardly played a role in state-of-the-art production plants and research concepts, since in most of them robots manage only a fixed and known number of operations within a production cell. Therefore, the limited number of utilized sensors and actuators allows for designing the quick-change system exactly according to the requirements. An example of this is shown in [2]. In the hybrid cell, subtractive (milling) and additive manufacturing processes (laser cladding) as well as laser scanning for quality control are combined in one robot cell. The end effectors required for this are integrated with quick-change systems. Here, sharing physical wires among communication protocols is not necessary due to the limited number of sensors and actuators.

With increasing use of robots, more complex end effectors are realized. In [3] an end effector for the fabrication of CFRP-based joints is presented. The end effector is controlled by several stepping motors, which communicate with the leading PLC of the robot cell via CANopen.

The use of dynamic partial reconfiguration (DPR) in communication systems has been investigated by Viswanathan et al. [4] and Dunkley [5]. Both suggest the use of DPR to efficiently implement different communication interfaces in a computation platform, given that these interfaces are not used at the same time. Dunkley aims for reducing the number of I/O interfaces needed to support various communication protocols in test equipment. This is achieved by connecting a DPR-capable FPGA to a computer system via PCIe. A reconfigurable region within the FPGA hosts the communication logic, while the static part is used to manage the PCIe communication to the PC. The setup is demonstrated using a simple RS232 example as well as a complex example including data processing in the reconfigurable region. However, the physical layer, i.e. the transceivers, is not considered. Targeting hardware obsolescence in avionics applications, Viswanathan et al. present a communication unit architecture that makes use of DPR to switch between different communication buses. In a purely FPGA-based setup, DPR is used to exchange different protocol implementations, while the main application runs on an FPGA-hosted CPU. CAN and the avionics communication protocol ARINC429 are implemented and integrated with an image processing application making use of the FPGAs capability to host hardware accelerators. Communication protocol hardware is connected to the FPGA using an FPGA Mezzanine Card (FMC) connector and thereby exchangeable. However, no information is given on the actual signal lines used for communication. Thus, the issue of limited signal lines is not discussed. In contrast, this work presents a hardware architecture concept targeting future robot-based industrial production systems, leading to the challenge of deploying industrial communication protocols over a fixed set of communication lines while supporting both current and future protocols.

3. DPR-based sensor integration concept

3.1. Requirements and relevant protocols

Considering the end effector reconfiguration process of a WSK robot, the maximum acceptable reconfiguration time of the sensor integration unit results from the usual time it takes to change a robot's end effector. In contrast to the tool changing system of conventional machine tools, changeover on the robot will be much slower. The entire process, which includes moving to the end effector magazine, depositing the end effector, picking up the new end effector and finally moving back to the operating position, takes at least one second and should not be delayed by the reconfiguration time of the sensor control unit.

Regarding communication protocols that are typically used in robot end effectors, among the large variety of sensors and actuators, communication often takes place via simple digital I/O signals, point-to-point communication interfaces, and field buses. According to an HMS Networks study on industrial net-

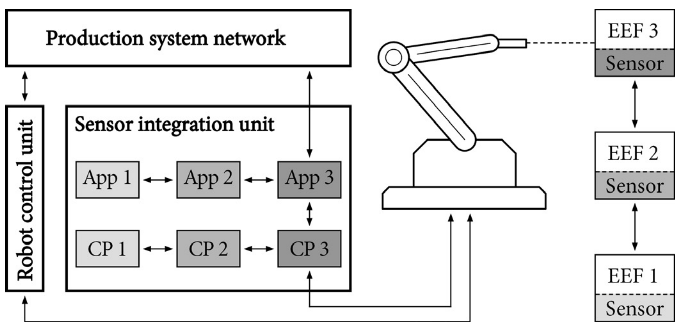

Fig. 1. System architecture of a WSK robot cell, including the proposed sensor integration unit

work market share growth in 2021, communication protocols such as Modbus-RTU, CC-Link, CANopen, and DeviceNet are, next to Profibus, very prominent among field buses [6]. Higher integrated sensors meanwhile mainly use Ethernet-based protocols like Profinet, EtherCAT, EtherNet/IP, or Modbus-TCP. Although the market share of Ethernet-based protocols is continuously growing, there still remains a large stock of field bus protocols used in the industrial environment [7].

To determine suitable protocols for the evaluation of our concept, we discuss the aspects in which industrial communication protocols and communication stacks typically differ. With regard to the physical layer (PHY) of the OSI reference model [8], these include the number of signal lines, voltage-based versus current-based signaling, signal voltage levels, line termination, and signal forms. These aspects are mainly realized in hardware using protocol-specific PHY ICs, also referred to as transceivers, and termination circuits. With regards to the data link layer, protocols may differ in aspects such as frame formats, synchronization, arbitration, and error correction, among others. Since these aspects can be realized at logic-level voltages, such functionality may be implemented in MAC ICs and is often integrated in embedded microcontrollers and systems-on-chip (SoC). Handling of higher-level aspects is mainly implemented in software.

Several industrial communication protocols use the same physical layer and, partially, data-link layer protocols. Examples include CAN, which is used in CANopen and DeviceNet or RS-485 used in Modbus-RTU and CC-Link. For that reason, this work focuses on physical- and data-link-layer protocols such as CAN and RS-485 for the evaluation of our concept.

3.2. *End effector sensor integration*

Figure 1 gives an overview of the architecture of a single WSK robot cell. Each robot module consists of the robot itself, a magazine of end effectors (EEF) comprising the tools and sensors, and a robot control unit, controlling the robot and tool operation. The robot control unit is connected to the production system network for management and synchronization. The production system network connects the robot modules to the overlying manufacturing execution system as well as to edge units for sensor data processing and forwarding.

To enable Industry 4.0 functions such as process monitoring, predictive maintenance, and to enable automatic quality control steps in robot-based production systems, the sensors integrated with the end effectors need to be interfaced and their data provided to the edge units for processing and storage. As discussed in Section 1 and Section 3.1, industrial sensors use a variety of communication protocols for configuration and data transmission. At the same time, the electro-mechanical interface between the robot arm and the end effectors is static and unified to enable exchangeability of end effectors. This leads to the challenge of interfacing different sensors over a fixed set of signal lines, which should be minimal to reduce costs.

A straightforward solution to this challenge is to add protocol conversion components to the robot end effectors, translating the sensor protocol into a common one used across the end effector interface. While this approach is certainly viable for many protocols, it has limitations: Given that end effector weight and cost of end effectors shall be minimized, integrating protocol converters adds overhead in both aspects. Additionally, by defining a fixed protocol across the quick-change system, the flexibility to upgrade to future protocols or to use more suitable application-specific protocols is lost. Therefore, we propose an approach where different protocols are supported over a fixed end effector interface.

To retain flexibility concerning the protocols used across the end effector interface, an adaptive, modular *sensor integration unit* is added, located at the robot control. By switching between different protocol implementations, it enables interfacing alternating sensors over a static set of communication lines. At the same time, it is connected to the production system network. Thereby, sensor-specific applications (App) and communication protocol (CP) implementations can be loaded in synchronization with the end effector attached to the robot for local sensor data processing or data forwarding to edge devices.

3.3. *Sensor integration unit architecture*

To create a sensor integration unit that is adaptable to different communication protocols in an efficient and future-proof way, the proposed architecture is based on a SoC combining a processor subsystem and an FPGA, coupled with an extension board hosting PHY ICs for the needed protocols and a multiplexer unit to dynamically assign the signal lines. The architecture is depicted in Figure 2 and is explained in the following.

The SoC's CPU is used to host sensor processing software and a management application on top of an operating system (OS). The management application controls the reconfiguration of the platform to switch between communication protocols. The FPGA is used to implement the data link layer specifics of the required communication protocols. More specifically, all protocol-specific implementation is placed in a partially reconfigurable region (PRR) in the FPGA, which can be updated at runtime without affecting other elements of the FPGA design. This allows application developers to use the remaining part of the FPGA for other purposes, such as hardware-accelerated sensor data processing. At the same time, this optimizes FPGA resource usage since only one protocol implemen-

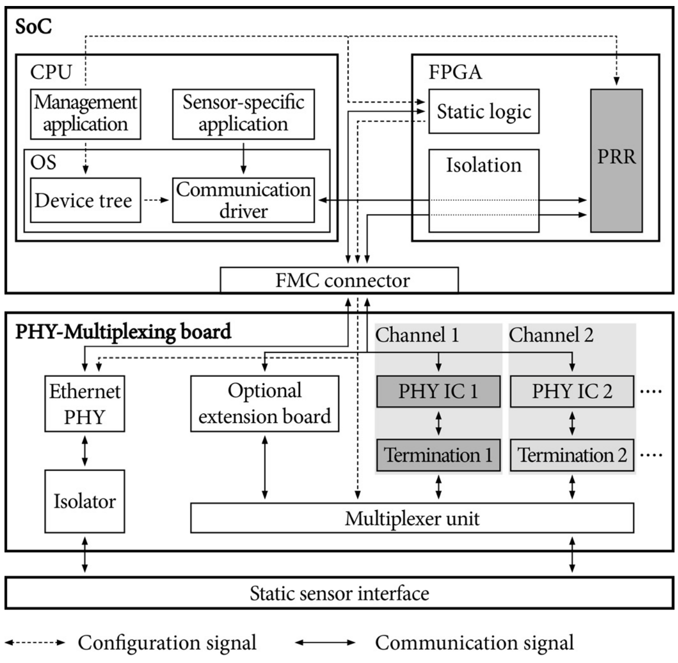

Fig. 2. Hardware/software architecture of the proposed sensor integration unit

tation needs to be loaded at a time, while unused implementations do not occupy FPGA resources and energy. To prevent erroneous output during reconfiguration, an isolation unit is located in the static FPGA design, decoupling the PRR during reconfiguration to guarantee system stability.

The low-level communication hardware required to realize the physical layer of the communication is located on an extension board, referred to as *PHY multiplexing board*. The PHY multiplexing board is organized in *channels*, each comprising the necessary hardware for a single communication protocol. A channel typically consists of the PHY IC and line termination circuitry as required by the protocol. In addition to the channels, the board hosts power supplies for all components. To simplify the integration of further protocols, channels can be realized on extension boards to the PHY multiplexing board.

Towards the FPGA, the PHY multiplexing board is connected via an FPGA Mezzanine Card (FMC) connector as defined in the ANSI/VITA 57 standard [9]. The FMC connector is specifically designed to allow for the extension of FPGA boards using daughter cards, thus simplifying the process of either exchanging the computation board or the extension board as necessary. Via this FMC interface, the PRR, containing the protocol data-link layer implementation, interfaces the corresponding channel.

Towards the sensor interface, i.e. the sensor signal lines, the multiplexer unit is located, comprising an array of analog multiplexers to allow for an arbitrary mapping of signal lines to channels. These multiplexers need to be specifically selected to meet the requirements of all protocols which need to be supported. This includes the maximum signal frequency, voltages, as well as cross-talk and signal distortion requirements. The multiplexer control signals are generated in the static part of the FPGA design and controlled by the management application. While the maximum frequencies supported by typical analog multiplexers may suffice to forward traditional industrial bus protocols, they may not be suitable for high-data-rate, Ethernet-based protocols. Therefore, a separate Ethernet channel is included on the PHY multiplexing board. Bypassing the multiplexer, it consists of an Ethernet PHY IC and isolator. It is interfaced by an appropriate MAC implementation in the static part of the FPGA design.

When multiple sensor protocols shall be supported in parallel, this architecture allows for the use of multiple channels in parallel. In this case, multiple reconfigurable regions are needed in the FPGA, hosting the data link layer implementations of the active protocols in parallel. This can be achieved by defining multiple PRRs. However, changing the number of PRRs at runtime requires a full re-programming of the FPGA, disturbing other FPGA implementations running in parallel.

The sensor integration unit hardware is controlled by a management application running on the CPU. It can be triggered by an overlying manufacturing execution system when the robot end effector is changed and then manages the sensor protocol reconfiguration process. By configuring the multiplexers and isolation units it ensures that no erroneous data is being sent during reconfiguration. It then loads the new protocol implementation design into the PRR. After reconfiguration of the FPGA it activates the operating system drivers needed to interact with the communication hardware by updating Linux device tree overlays. These drivers can then be used by sensor-specific applications to communicate to the sensor.

4. Prototypical implementation and evaluation

4.1. System implementation

The concept was implemented based on an Avnet ZedBoard, a development board for Xilinx Zynq-7000 SoCs, for which a custom PHY multiplexing board was designed. Based on the considerations discussed in Section 3.1, CAN and RS-485 bus communication interfaces were realized as well as a digital I/O interface and a means to capture analog signals. Also, the non-multiplexed Ethernet interface was implemented.

Consisting of a dual-core ARM Cortex-A9 processor and a Xilinx FPGA, the Xilinx Z-7020 was used to implement the SoC component of the concept. As presented in Section 3.3, the FPGA design comprises a static part and a partially reconfigurable region, which was defined to match one of the FPGA's clock regions. In the static part, General-Purpose I/O (GPIO) peripherals were instantiated to control the multiplexers on the PHY multiplexing board as well as decoupling logic to isolate the PRR. For the partially reconfigurable region, four different MAC layer implementations were created. The CAN MAC implementation was realized using Xilinx AXI CAN IP core. For RS-485, a UART (universal asynchronous receiver/transmitter) component was used along with a GPIO module to control the transceiver operation. For digital inputs and outputs, the reconfigurable region consists of a GPIO module. To receive analog current and voltage signals, another MAC-implementation

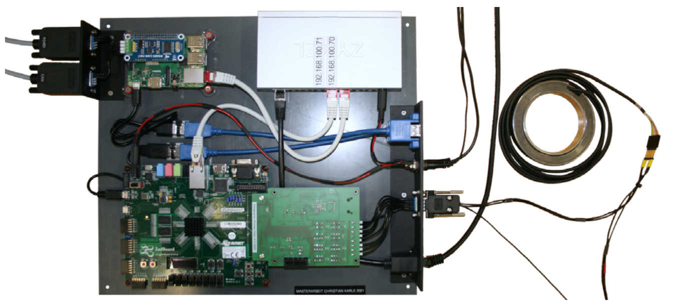
Fig. 3. Implementation prototype with CAN-based spindle sensor attached

was created, interfacing the analog digital converter on the PHY multiplexing board extension.

On the PHY multiplexing board, channels for CAN and RS-485 were designed. The multiplexing unit was realized by several analog multiplexers, creating a configurable, bi-directional connection between one of eight sensor lines and the channels' termination circuitry and PHY ICs. The required PHY-ICs for CAN and RS-485 were placed directly on the PHY multiplexing board. Extensibility was achieved by defining one channel where signals coming from the SoC as well as signals coming from the multiplexers are forwarded to a PCIe connector. Thereby, PHY implementation and termination circuitry can be defined by minimal, exchangeable add-on boards. To test this channel, a compatible extension board was designed, hosting the analog digital converter for analog signal capturing. To allow for the integration of Ethernet-based sensors, an Ethernet PHY with a corresponding isolation component was added on the PHY multiplexing board.

The software running on the SoC CPU is based on an embedded Linux distribution developed using PetaLinux. On top of it, a management application was developed, automating the protocol reconfiguration tasks that arise during the system configuration when changing between the different bus systems. To allow for low-latency access to the modules instantiated in the FPGA, it uses direct memory mapping of the respective components' register space. Changes to the Linux device tree are made using the file system interface.

4.2. Test and evaluation

The implementation was tested regarding simple digital inputs, CAN and RS-485 communication, and evaluated concerning resource usage and reconfiguration latency. Additionally, the system was tested using an industrial sensor.

To test the CAN and RS-485 communication, a Raspberry Pi using an RS-485+CAN extension board was attached to the sensor interface. RS-485 communication was configured at 115.2 kBd and CAN communication at 500 kbit/s, transmitting artificial sensor data generated by the Raspberry Pi to the sensor integration unit, where it was visualized by a Linux application, which also controlled the protocol reconfiguration. To verify real-world usability, the system was furthermore tested using the industrial spindle monitoring sensor SpindleSense by Schaeffler which communicates by high-speed CAN

Table 1. Reconfiguration latency when loading the CAN implementation, 1000 measurements

Reconfiguration step	**Latency [µs]**	
	mean ± SD	***max***
Reset multiplexer	10.3 ± 0.785	11.8
Unload device tree overlay	96 639 ± 14 814	166 230
Decouple FPGA region	16.4 ± 0.764	28.1
Load bitstream	132 770 ± 390	134 330
Couple FPGA region	2.70 ± 0.102	3.08
Load device tree overlay	4512 ± 246	6925
Configure multiplexer	16.4 ± 0.500	18.4

Table 2. FPGA resources occupied by the implementation on the Xilinx Z-7020

Design	**Lookup tables**	**Registers**	**Multiplexer**
Static	15 687 (29.7 %)	21 098 (20.0 %)	30 (< 0.1 %)
CAN partial	1040 (2.0 %)	917 (0.9 %)	12 (< 0.1 %)
RS-485 partial	728 (1.4 %)	622 (0.6 %)	1 (< 0.1 %)
GPIO partial	512 (1.0 %)	405 (0.4 %)	0 (0 %)

at 1 Mbit/s [10]. This test setup is depicted in Figure 3. The suitability for digital signals was tested by reading a digital input signal issued by a hall sensor. In a demo application this was used to determine the rotational speed of a wheel. In all cases, the sensor data was received correctly.

The evaluation of reconfiguration latencies was conducted by augmenting the management application, adding time measurement code to all reconfiguration steps. 1000 latency measurements were recorded during the reconfiguration of the system for each protocol implementation. Table 1 shows the resulting latencies for the partial configuration of the CAN protocol. For the RS-485 and GPIO implementations, similar latencies could be seen except for the device tree operations. Unloading and loading the RS-485 device tree overlay resulted in mean latencies of 2050 µs and 29 961 µs, respectively. For the GPIO implementation, this resulted in mean latencies of 937 µs and 4115 µs. These differences originate from protocol-specific Linux driver behavior during loading and unloading. This resulted in mean overall reconfiguration latencies of 234 ms for the CAN implementation, 165 ms for RS-485, and 138 ms for the GPIO design. Note that the initialization of memory mapping to control the decoupling unit and multiplexers is not included in the measurements since it needs to be done only during the startup phase of the management application.

The FPGA resources occupied by the static design and the partial designs for different protocols are shown in Table 2. It can be seen that the static design takes about 30 % of the Z-7020's lookup tables and 20 % of its registers. The multiplexer resource usage is minimal. It should be noted that not all remaining resources can be used for other designs. Since reconfigurable regions are statically defined and need to be sized based on the largest partial implementation, unused resources within them may not be used by other top-level design components. However, the partial design may be extended to contain additional functionality besides the communication logic.

5. Discussion

Based on the requirements derived in Section 3.1 and the evaluation results in Section 4.2, it can be seen that the presented implementation meets the reconfiguration latency requirements for the investigated protocols. Furthermore, the evaluation shows that a significant number of the industrial communication protocols, i.e. the CAN-based and RS-485-based protocols, can be realized on the presented platform. Thereby, the number of signal lines at the robot end effector interface can be kept low even when multiple protocols are used. Compared to implementing such protocols over dedicated lines, this can reduce the size and in some cases the number of feed-through modules needed, lowering production system costs.

While Ethernet-based protocols are supported by the platform as well, they cannot be reconfigured at runtime. Instead, they can be used directly via the Ethernet interface of the FPGA board or the Ethernet channel on the extension board. Concerning the end effector interface, this means that Ethernet communication lines will need separate signal lines alongside the reconfigurable lines.

In general, the presented concept may be applied to various protocols. In practice, applicability is limited by several factors such as the maximum signal frequencies that need to be transmitted to conform to the protocol specification. In our implementation, the analog multiplexer has a bandwidth of approx. 28 MHz, preventing the use of higher-frequency protocols. Another limiting factor can be requirements towards cabling characteristics such as impedance and shielding. While the termination circuitry is part of the extension board channel and thereby can be realized in a protocol-specific way, the cables within the robot are, by definition, the same for all protocols. Hence, appropriate cabling is necessary to ensure suitability to all protocols that are to be implemented. Therefore, where two protocols have conflicting requirements with regards to cabling, they may have to be routed through separate cables.

6. Conclusion

Targeting future, robot-based production system architectures such as the Wertstromkinematik concept described in [1], we identified the challenge of including varying smart end effectors comprising different sensors into the production system in a future-proof and efficient way, while keeping cost and weight of said end effectors low. To face this challenge, we proposed an architecture which allows for flexibly alternating between different communication protocols over a static set of signal lines to achieve a simple and lightweight electro-mechanical interface towards end effectors. Exploiting the dynamic partial reconfiguration feature of today's FPGAs to realize updateable communication logic, we created an efficient sensor integration unit architecture. We presented a prototypical implementation showing its feasibility for CAN- and RS-485-based communication protocols as well as digital I/O lines and evaluated it regarding resource usage and reconfiguration latencies. We found that the observed latencies render the concept feasible for the desired application and discussed advantages and limitations when implementing further protocols.

The proposed architecture can, in principle, also be used to control complex actuators in robot end effectors. However, this application was not explicitly investigated. Hence, an evaluation of this use case is considered future work.

Acknowledgment

The authors would like to express their appreciation to Karlsruhe Institute of Technology for its Future Fields Funding program supporting the "Wertstromkinematik" project.

References

[1] E. Mühlbeier, P. Gönnheimer, L. Hausmann, J. Fleischer, Value Stream Kinematics, in: B.-A. Behrens, A. Brosius, W. Hintze, S. Ihlenfeldt, J. P. Wulfsberg (Eds.), Production at the leading edge of technology, Springer Berlin Heidelberg, Berlin, Heidelberg, 2021, pp. 409–418. doi:10.1007/978-3-662-62138-7_41.

[2] C. Baier, F. Hähn, C. Tepper, M. Weigold, Robot-based hybrid production concept, in: J. P. Wulfsberg, W. Hintze, B.-A. Behrens (Eds.), Production at the leading edge of technology, Springer Berlin Heidelberg, Berlin, Heidelberg, 2019, pp. 451–460. doi:10.1007/978-3-662-60417-5_45.

[3] M. Dackweiler, Modellierung des Fügewickelprozesses zur Herstellung von leichten Fachwerkstrukturen, Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2020). doi:10.5445/IR/1000122295.

[4] V. Viswanathan, B. Nakache, R. Ben Atitallah, M. Nakache, J.-L. Dekeyser, Dynamic reconfiguration of modular I/O IP cores for avionic applications, in: 2012 International Conference on Reconfigurable Computing and FPGAs, 2012. doi:10.1109/ReConFig.2012.6416741.

[5] R. Dunkley, Supporting a wide variety of communication protocols using dynamic partial reconfiguration, IEEE Instrumentation Measurement Magazine 16 (4) (2013) 26–32. doi:10.1109/MIM.2013.6572950.

[6] HMS Networks, Continued growth for industrial networks despite pandemic, https://www.hms-networks.com/news-and-insights/news-from-hms/2021/03/31/continued-growth-for-industrial-networks-despite-pandemic, Accessed: 2021-11-30 (2021).

[7] Hilscher, Single-pair Ethernet: Market Potential and Hilscher's approach, https://www.hilscher.com/blog/single-pair-ethernet-market-potential-and-hilschers-approach/, Accessed: 2021-11-30 (2019).

[8] Information Technology – Open Systems Interconnection – Basic Reference Model, Standard, International Telecommunications Union (1994).

[9] R. Seelam, I/O design flexibility with the FPGA mezzanine card (FMC), Xilinx White Paper WP315 (2009).

[10] Schaeffler Technologies AG & Co. KG, Schaeffler SpindleSense (TPI 258), https://www.schaeffler.de/remotemedien/media/_shared_media/08_media_library/01_publications/schaeffler_2/tpi/downloads_8/tpi_258_de_en.pdf, Accessed: 2021-12-09.