The Computing and Communication Architecture of the DLR Hand Arm System

Stefan Jörg and Mathias Nickl, Alexander Nothhelfer, Thomas Bahls and Gerd Hirzinger

***Abstract*— The computing and communication architecture of the DLR Hand Arm System is presented. Its task is to operate the robot's 52 motors and 430 sensors. Despite that complexity, the main design goal for it is to create a flexible architecture that enables high-performance feedback control with cycles beyond 1kHz. Flexibility is achieved through a hierarchical net of computing nodes that goes from commercial-of-the-shelf hosts down to the physical interfaces of sensors and actuators. The concept of a Hardware Abstraction Layer (HAL) provides a convenient high-level interface to the entire robotic hardware. First experiments with prototypical control applications, featuring 100 kHz and 3 kHz control loops, demonstrate the performance of the architecture.**

I. Introduction

The DLR Hand Arm System (see Fig. 1) is an anthropomorphic system that is aimed to reach its human archetype regarding size, weight and performance. It features intrinsic compliance implemented as variable stiffness actuation [1]. The hand arm system has in total 26 DOF, thereof 19 DOF in the hand, 2 DOF in the wrist, and 5 DOF in the arm. To implement all those DOF, the hand arm system comprises 52 actuators and 430 sensors of different types (see Table I).

To operate that many actuators and sensors precisely for a certain control application the complexity of the system needs to be hidden from application designers. On the other hand, in order to maintain good performance the application must have the most direct access to all actuators and sensors. In other words, a valuable means of abstraction with only minimal execution overhead is required.

This is the task of the *Computing and Communication Architecture*. It incorporates the operating software and the computing and communication infrastructure of the DLR Hand Arm System. The aim is to provide a convenient high-level hardware abstraction that still allows high-performance feedback control with cycles beyond 1kHz.

Related Research

Designing computing and communication platforms is a challenging task for every humanoid robotic project. Especially, the large number of sensors and actuators, the level of integration, sample-rate and latency, mechanical constraints, and even the project funding affect the design of computing platforms.

All humanoid computing and communication architectures are complex compositions of computing nodes (e.g. CPUs, DSPs, or FPGAs), buses (e.g. Ethernet, CAN, PCI), as well as sensors and actuators. Most platforms are implemented with standard hardware components, originated in the automation industry. Some projects develop dedicated electronic hardware, e.g. motor drivers, to reach a higher level of integration.

The HRP-II [2] by AIST, the Wabian-2 [3] by the University of Tokyo, and JOHNNIE [4] by Technische Universität München use PC technology extended by PCI-I/O cards.

All authors are affiliated with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany,
e-mail: stefan.joerg@dlr.de

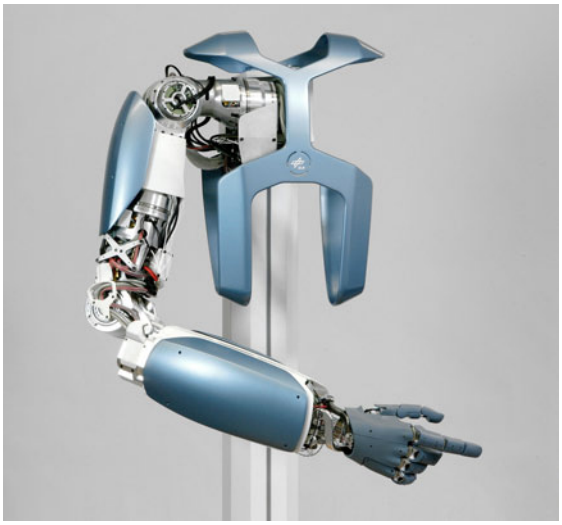

Fig. 1. The DLR Hand Arm System

part		Motors		Sensors
Arm	4	RoboDrive Ilm50	4	Magneto-Resistive
	6	RoboDrive Ilm25	6	Angular Hall enc.
			15	Potentiometers
			30	Current
			6	Voltage
			25	Temperature
Wrist	4	RoboDrive Ilm25	8	Angular Hall enc.
			3	Potentiometers
			12	Current
			4	Voltage
			9	Temperature
Hand	38	RoboDrive Ilm25	76	Angular Hall enc.
			114	Current
			38	Voltage
			80	Temperature
Total	52		430	

TABLE I

The Motors and Sensors of the Hand Arm System

KHR-2 [5] of KAIST and the HRP-3 [6] of AIST combine a PC-platform with CAN bus communication.

HONDA's ASIMO [7] consists of PCs, DSPs, and PCI I/O cards as well as PCI and Ethernet for backplane communication.

The PR2 [8] by Willow Garage is a hierarchical platform with CPU and Motor Control Boards that are connected by EtherCat. ARMAR-III [9] by Universität Karlsruhe is a hierarchical platform with PC104-PCs and integrated DSP-FPGA boards that are connected by Gigabit Ethernet, and CAN to connect sensors and actuators.

LOLA [10] by Technische Universität München is based on PC technology and Sercos-III to connect sensors and actuators. DLR's Justin [11] is a heterogeneous platform with 4 PCs, Motion Controllers, SERCOS, EtherCat, Gigabit Ethernet, and CAN. NASA's Robonaut 2 [12] consists of distributed FPGA-PowerPC based motor controllers that are connected by a custom communication to a central Compact-PCI PowerPC based host.

Computing platforms have evolved from monolithic to distributed control platforms. In this course new platform concepts tackle the therefore necessary distribution of control algorithms. Despite these efforts, the seamless distribution of high-bandwidth control algorithms to heterogenous platforms is still a challenge.

Above the operating system level, robotic middleware frameworks handle distribution. Some frameworks have the focus on hard real-time control loops. DLR's aRD [13] uses UDP communication and supports QNX, Linux and VxWorks. DDS [14] is a service-oriented real-time middleware used by NASA's RAPID [15] project.

Other robotic middleware frameworks are flexible and extensible component frameworks that have the focus on a more abstract command level. ROS [16] by Willow Garage provides a structured communications layer above host operating systems. MCA2 [17] handles distributed components for Linux with RTAI/LXRT extensions. OpenHRP [18] is a high-level CORBA based simulator and motion control library.

The DLR Hand Arm System's dedicated and heterogenous platform is below common operating system interfaces. In particular, common robotic middlewares do not support FPGA or Micro-Controller platforms. Moreover, the goal of control loops beyond 1kHz requires hard real-time support. The hand arm system's distributed, dedicated hardware asks for a distributed but yet deterministic device driver.

The presented solution combines the concepts of device drivers and middleware. It has the focus on the control application level with hard real-time constraints and not on a more abstract command level.

This paper is organized in three parts. The following two sections present the computing and communication architecture and the electronic hardware components. Sec. IV, *The Hardware Abstraction Layer*, outlines the robot's operating software. Finally, Secs. V and VI present the results of first control application implementations.

II. The Computing and Communication Architecture

To balance the opposing requirements of flexibility and high integration, the DLR Hand Arm System's computing and communication platform is laid out hierarchical: At the top are general purpose, commercial-of-the-shelf (COTS) components. The footprint decreases towards the bottom end which is defined by the dedicated physical interfaces of sensors and motors. The available computing power and communication bandwidth decreases along with the decreasing footprint. A modular layout on each level (see Sec. IV) together with the aggregation of components on successive levels by the means of suitable communication creates the desired platform flexibility (see Fig. 2). This hierarchy is not driven by a functional separation but only by the requirement of small footprint sizes at the physical interfaces. The functionality of an application can be flexibly mapped onto this hierarchy as required.

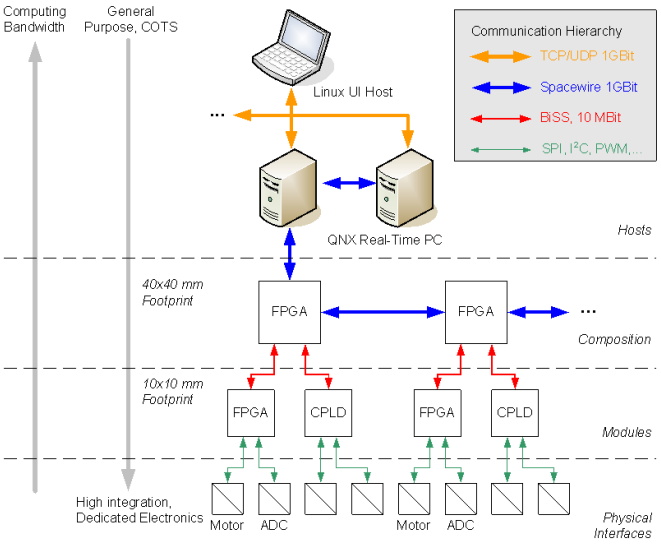

Fig. 2. The DLR Hand Arm System's hierarchical computing and communication platform has four layers of integration scale for computing and communication.

A. The Computing Hierarchy

The top layer of the computing hierarchy, the *Host Layer*, are COTS PCs running the real-time OS QNX. On this layer run control applications that use a Simulink RTW development tool chain. The real-time hosts are augmented by auxiliary Linux workstations intended for user interfaces. Below this, the *Composition Layer* is constituted by FPGAs of the XILINX Virtex 5 family (package class 40mmx40mm) which provide powerful computing nodes. The Composition Layer aggregates the many modules of the lower level to subsystems and systems . The *Module Layer* is the lowest computing layer with small footprint CPLDs and FPGAs (package class 10mmx10mm). Its task is to integrate the dedicated parts of the physical interfaces with their proprietary communication into the hierarchy and to provide modules

with a common communication. The *Physical Interfaces* constitute the most dedicated components of the architecture. Specific ADC parts and power inverters are the immediate interface to the analog physical world.

B. The Communication Hierarchy

The applied communication protocols follow the same hierarchy as the computing nodes: At the top, the general-purpose platforms are connected by high-bandwidth standard communication protocols. Close to the physical interfaces, low-bandwidth protocols are employed that feature small implementation footprints.

The auxiliary Linux hosts are connected to each other and the QNX real-time hosts via standard 1GBit TCP/UDP. SpaceWire is the backbone of the *Composition Layer* that connects the QNX real-time hosts to the FPGA computing nodes on this layer. SpaceWire is a low footprint packet based bus that is deterministic for a given topology [19]. The physical layer is implemented with an IEEE 802.3 Gigabit Ethernet compliant Ethernet transceiver featuring 1GBit/sec bandwidth. The FPGA/CPLDs of the *Module Layer* are connected to the composition nodes with the industry standard BiSS C-Mode [20]. BiSS is a Master/Slave bi-directional serial bus with a typical data rate of 10Mbit/s. Matching the hierarchy, the slave's footprint is much smaller than the master's. The *Physical Interfaces* on the bottom are connected to the module layer with their dedicated communication protocols: SPI, I2C buses or Pulse-width modulation (PWM) outputs.

The flexibility of the DLR Hand Arm System's computing and communication architecture is provided by programmable devices on the two intermediate Composition and Module layers. Available computing power all the way down to the physical interfaces opens up flexibility for both computing and communication. Both, the SpaceWire and BiSS protocols are entirely implemented in VHDL software on the FPGA/CPLD computing nodes. Thus, both protocols may be replaced by any communication protocol that fits to the provided serial physical layers.

III. The Electronic Hardware

In the following section a short explanation of the key electronic components (see Fig. 3) in matters of computing and communication of the DLR hand arm system is given. The design of the electronic components follow the hierarchy depicted in Fig. 2. Composition nodes are FPGA-based communication and computation platforms that support dedicated communication protocols, i.e. SpaceWire and BiSS. Modules are sensor and motor electronics, which convert the analog physical interfaces to dedicated communication protocols.

A *SpaceWire PCIe Card (C1)* connects the standard COTS PCs (see II) to the SpaceWire network, which implements the backbone of the composition layer. This card consists of an Xilinx Virtex5 FPGA and two fiber optical channels for SpaceWire communication. A dedicated fiber to copper transceiver (omitted in Fig. 3) allows the seamless connection of fiber and copper SpaceWire networks.

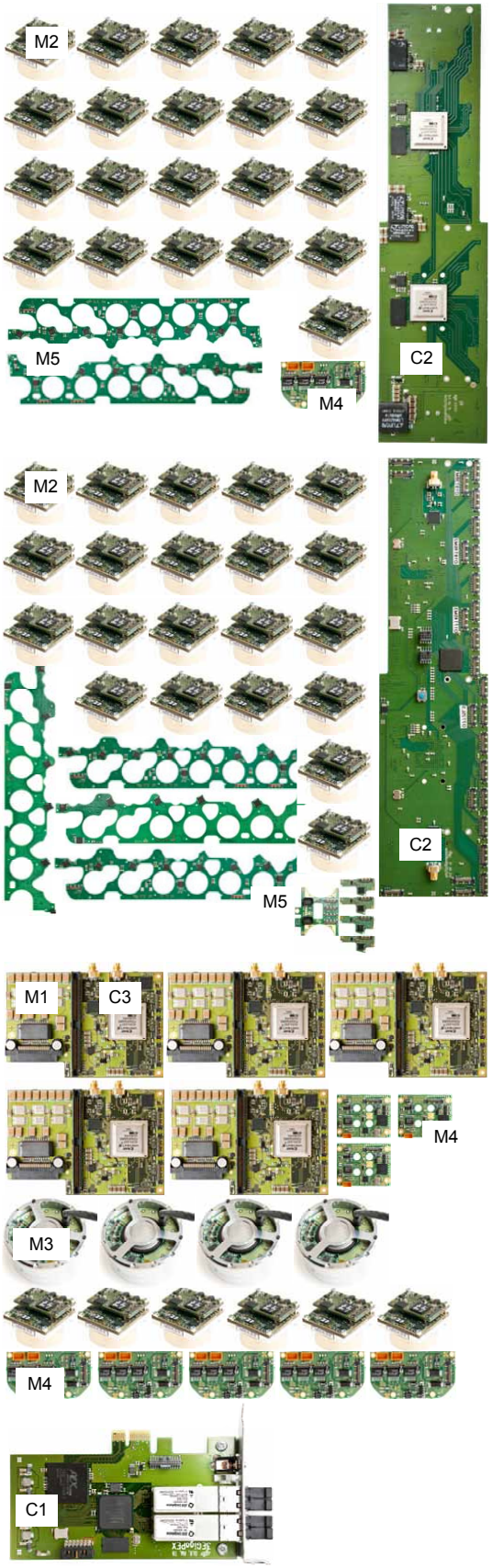

Fig. 3. The electronic components of the DLR Hand Arm System. *Composition Layer:* C1 SpaceWire PCIe Card, C2 Forearm Node, C3/M1 Arm Node, *Module Layer:* M2 ILM25 MiniServo, M3 ILM50 with Magneto-Resistive Sensor, M4 Potentiometer Module, M5 Angular Hall Modules

The two *Forearm Node (C2)* electronics comprise signal aggregation and routing. They are used as SpaceWire communication backbone and are capable of connecting 26 BiSS modules. Each of them has two Virtex5 FPGA (V5LX50) and provides 26 BiSS masters, the SpaceWire links and routers, and temperature monitoring.

The *Arm Nodes (C3/M1)* are stackable circuit boards with communication and computing capabilities combined with a fully integrated motor power inverter with three SPI current sensors and I2C temperature sensors. The module provides one Virtex5 FPGA (V5LX50) as computation resource, two interfaces with 1 $\frac{Gbit}{sec}$ SpaceWire by using IEEE 802.3 physical layer, 8 BiSS connectors, two interfaces with 100 $\frac{Mbit}{sec}$ for data exchange in between multiple stacked *Arm Nodes*.

The *ILM25 MiniServo (M2)* motor module is a very compact self-contained intelligent component, which consists of a PMSM ILM25, inverter electronics, and control electronics. A Xilinx Spartan 3e XC3S500EP132 with only 8x8 mm is responsible for communication (BiSS) as well as position and current control of the motor at 100 kHz. Control asks for three SPI current sensors and one angular hall position sensor with a resolution of $4096\frac{Incs}{rev}$, BiSS communication and a very small volume consumption. Furthermore, for housekeeping, two I2C temperature sensors and one SPI sensor for the DC link voltage are employed. The entire hand arm system uses 48 MiniServos in total.

A custom-designed *Magneto-Resistive Position Sensor (M3)* with a resolution of $23040\frac{\text{Incs}}{\text{rev}}$ is employed as position sensor for the main motors ILM50. This sensor is optimized for *hollow shaft drives*.

Potentiometer Modules (M4): In order to measure the deflection of the elastic elements a CPLD is used to read the digitized sensor data of the analog potentiometers and transfer data via BiSS to the superimposed composition nodes.

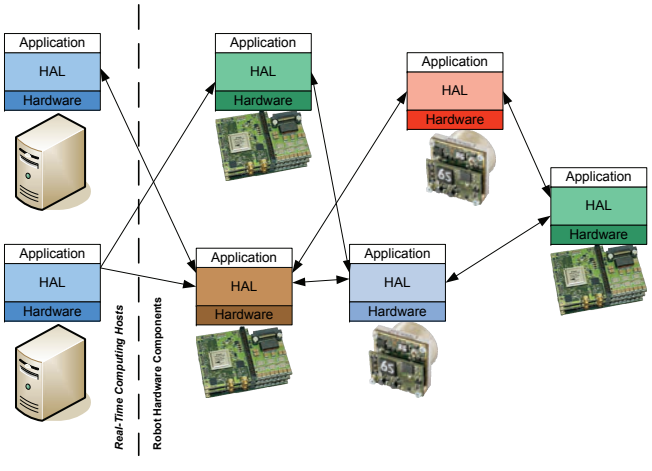

Fig. 4. The signal-oriented Hardware Abstraction Layer (HAL) binds the distributed hardware to the distributed application. For dedicated platforms (e.g. FPGA), the HAL reaches below the operating system layer.

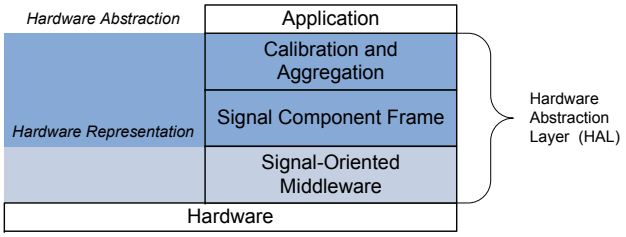

Fig. 5. The HAL is a signal-oriented component-based device driver concept. It tackles hardware distribution with a signal-oriented middleware that presents the hardware functionality as is: The Hardware Representation Layer. On top of this a signal-oriented component frame handles synchronization and scheduling. Data aggregation and calibration enhance the hardware functionality for user convenience.

Angular Hall Sensor (M5): The same type of angular hall position sensor that is utilized in the ILM25 MiniServos provides measurement of the tendon deflection in the hand as well as in the wrist.

IV. The Hardware Abstraction Layer

The *Hardware Abstraction Layer (HAL)* is a distributed device driver. The HAL is present on every node of the electronic hardware where a part of the application is running (see Fig. 4).

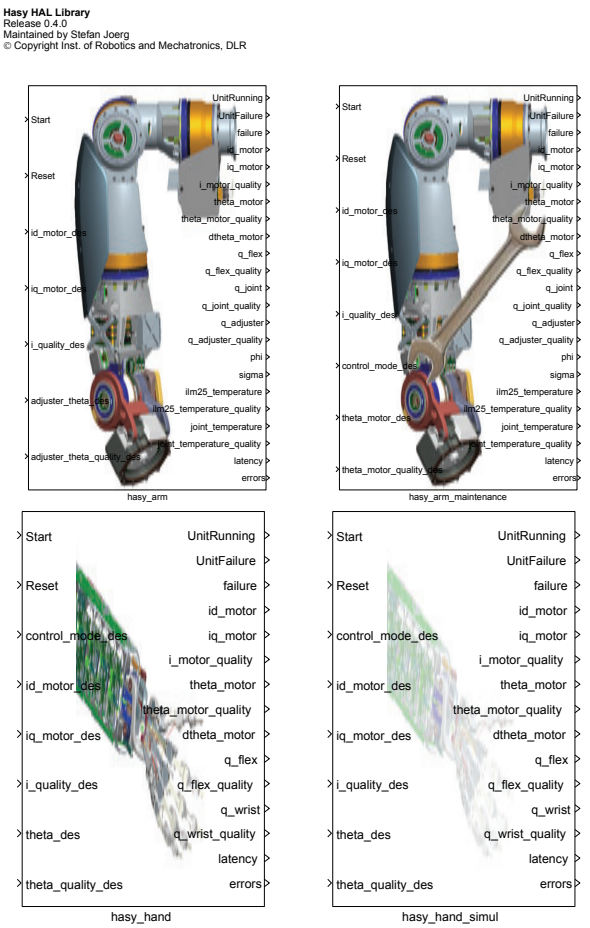

Fig. 6. The HAL Simulink block library presents the hardware as high level interfaces that feature calibrated sensor values in SI-Units.

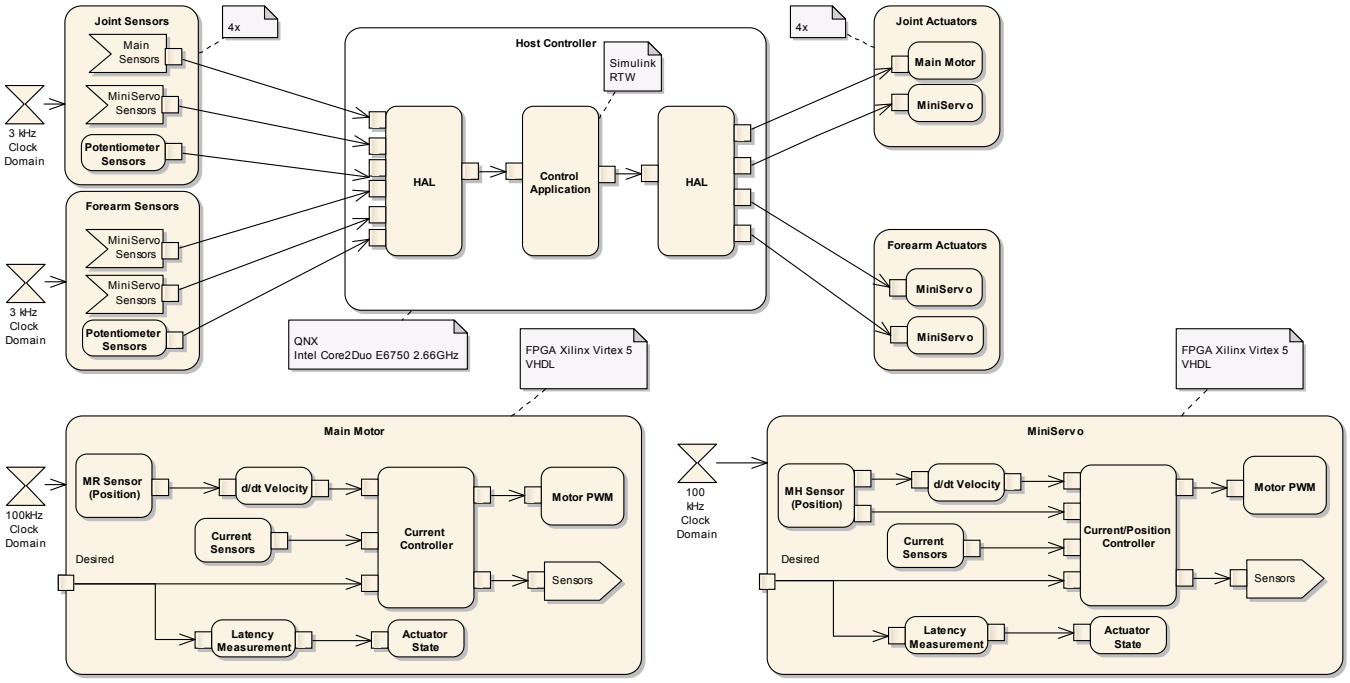

Fig. 7. Top: The arm component architecture drives 4 main motors, 4 adjuster MiniServos, and 2 forearm MiniServos. The HAL aggregates all sensors and motors to one block. Both, HAL and arm controller application run on a QNX real-time host. Bottom: The FPGA implementations of the main motor current controller (left) and the MiniServo current/position controller (right) both operate at 100 kHz.

The HAL provides an abstract view on the robot's distributed hardware. Therefore, the HAL hides all unwanted hardware details like communication protocols and sensor specific value types. It implements a convenient interface intended for the use by control designers that represents all values as floating point SI-units. This convenient interface follows what we call a *signal-oriented component model*, which is based on the actor model [21].

Fig. 5 illustrates the three tasks the HAL accomplishes in order to provide a convenient hardware abstraction:

The Signal-oriented Middleware

The DLR Hand Arm System's HAL uses the signal-oriented middleware concept presented by Joerg et al. [22]. The middleware handles the distribution of heterogenous hardware platforms including FPGAs.

The Signal-oriented Component Frame

The component model of the hand arm system's HAL is based on the formal *Synchronous Model* that assumes discrete signals with a fixed sample period [23]. The domain-model *The Virtual Path*, presented by Nickl et al. [24], is used for the implementation of the HAL's component frame. The model defines the four roles Sensor, Controller, Actuator, and Communication. It distinguishes *synchronization* in terms of synchronous to physical time and *scheduling*. *Sensors* are synchronized to physical time, i.e. a sensor is triggered by a tick, which is an event that is synchronous to the physical time. An *Actuator* has a watchdog clock, which is synchronous to physical time. It is not necessary to synchronize the controllers. The communication between controllers can be implemented as simple *FIFO channels*.

Calibration And Aggregation - The Simulink interface

The HAL implements a convenient interface on any computing node at which a part of the control application is executed. On platforms where a Simulink RTW tool chain is available the interface is implemented as Simulink block library. Thus control designers are able to interface their prototypical control applications with the DLR Hand Arm System's hardware. Fig. 6 depicts the two Simulink libraries for the hand and arm units. Every block represents a certain unit of the distributed robot hardware. For example, the 48 actuators and the corresponding sensors of the hand are represented by one single block. The aggregation of the concerned sensor value signals is implemented in the HAL. The interface of each block consists of floating-point SI-values that represent calibrated sensor values. The necessary calibration models are also implemented in the hardware abstraction layer.

V. First Control Applications

To conduct first control experiments, presented by Grebenstein et al. [1] and Petit et al. [25], a prototypical signal component architecture was implemented.

The control applications use a cascade control architecture: The inner control loop is the actuator controller which operates at 100 kHz and the outer host control loop operates at 3kHz. The latter includes the HAL Simulink interface and the host control application implemented with Simulink.

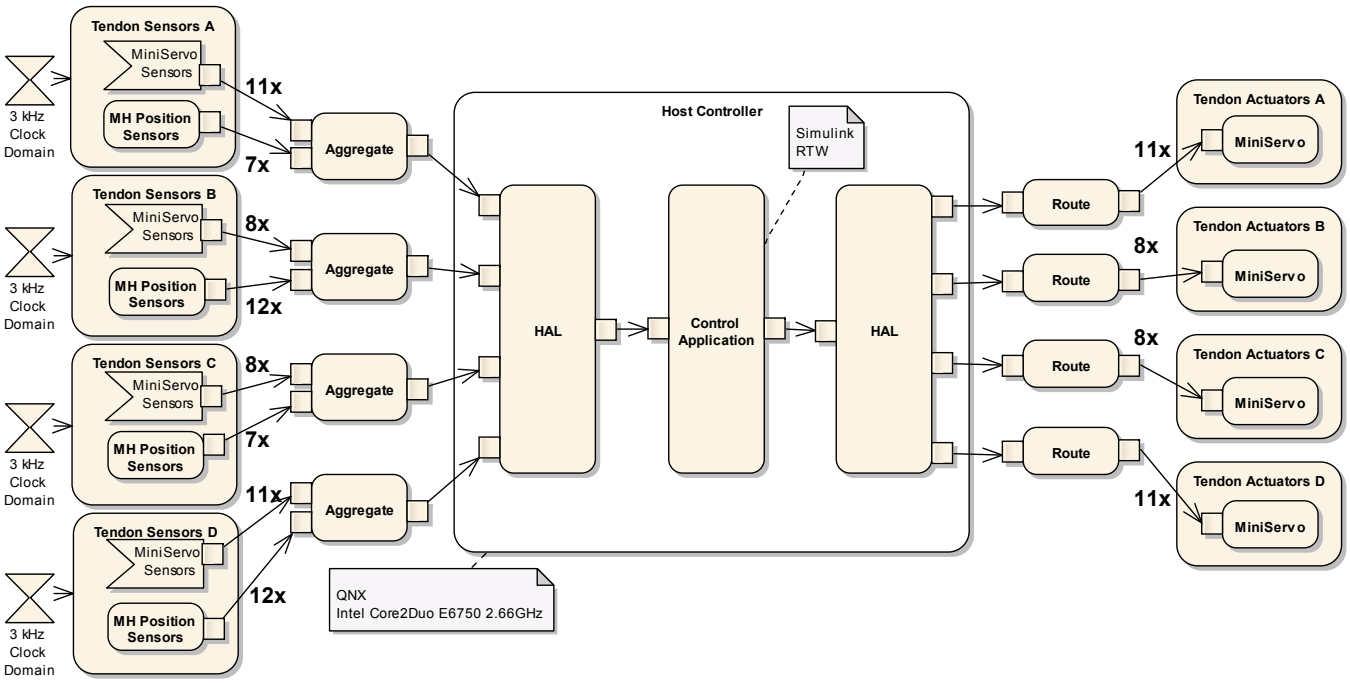

Fig. 8. The hand component architecture. The whole system is synchronized by its sensors (3kHz and 100kHz clock domains). 38 tendon MiniServo modules (see Fig. 7) are aggregated to one block by the HAL. Both, HAL and control application run on a QNX real-time host.

The former is implemented entirely on the joint FPGAs. All controllers are implemented with the *Virtual Path* model [24], i.e. the loops are synchronized to their clock domain (either 100kHz or 3kHz) by the sensors (current, position).

A. The Arm

The arm part of the DLR Hand Arm System consists of five joints. The first four joints are FSJ joints, presented by Wolf et al. [26], that consist each of one main RoboDrive Ilm50 motor and one stiffness adjuster MiniServo motor. The fifth joint, implementing the forearm rotation, consists of two MiniServos. This is reflected by the arm's signal component architecture illustrated in Fig. 7. The main motor's current control loop operates at 100kHz and is implemented on the FPGAs of the joint layer of the computing hierarchy. The same applies to the MiniServo implementation (see next section). The HAL aggregates all sensors and actuators and is implemented as a simulink block library (see Fig. 6). The state feedback damping control approach [25] is implemented with Simulink using the HAL block library. Both, HAL and control application run on a QNX real-time host.

B. The Hand

The hand comprises 19 DOF that are implemented as antagonistic tendon drives. Therefore 38 MiniServo modules (see Sec. III) are used. The position of each of the 38 tendons is measured by an angular Hall encoder. Fig. 8 depicts the signal component architecture for the hand. The MiniServos are arranged in four groups each of which is aggregated to one SpaceWire message on a FPGA computing node of the joint layer. The MiniServo has a current and a position controller. Both operate at 100kHz and are implemented on the FPGAs of the joint layer of the computing hierarchy. The outer host control loop operates at 3kHz. It forms a cascade control architecture together with the MiniServo's controller as the inner loop. The HAL is a part of the host control loop and aggregates all MiniServos and sensor interfaces for the finger impedance controller application, presented in [1].

VI. Results

In order to validate the goal of high-performance control, the performance of the implementation is evaluated by the experimental identification of the latency of the control loops. The *Virtual Path Model* [24] defines latency as the accumulation of all computation and communication delays of the virtual path from sensors to actuators. Hence, for the outer loop of the cascaded control architecture latency is

$$t_{Latency} = t_{SetDesired} - t_{tick3kHz}$$

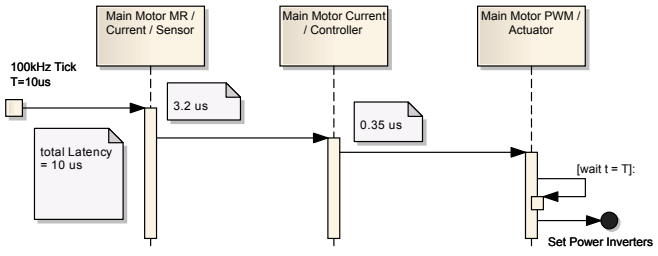

Fig. 9. The latency analysis of the 100kHz current control loop of the arm's main motors from Sensor ADCs to the Pulse-width modulation (PWM) output. The computing latency is $3.55\ \mu s$. The PWM is synchronized to the clock, i.e. the overall latency is exactly $10.0\ \mu s$.

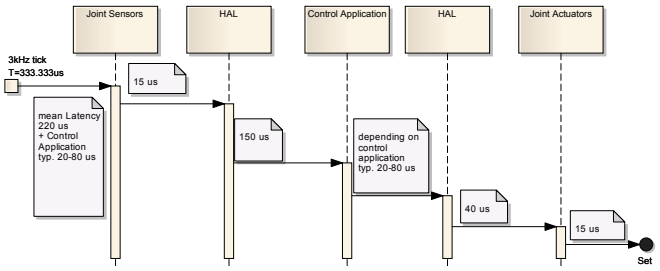

Fig. 10. Execution sequence of the arm's 3kHz host control loop. The communication latency averages to 30 μs, the HAL averages to 190 μs. The host controller computing time depends on the control application. See Fig. 11 for detailed measurements.

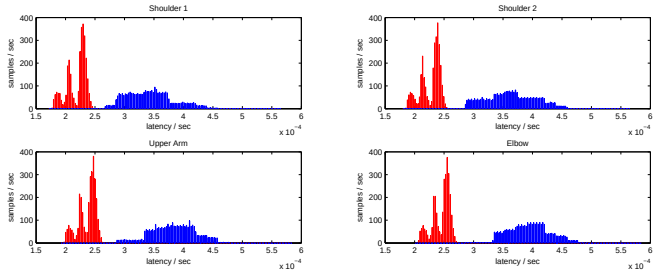

Fig. 11. The latency of the 3kHz control loops for the arm's four main motors. The histograms show the latency distribution for the damping control application (blue) and for HAL only (i.e. no host control application) (red).

To exactly measure the latency, the *Latency Measurement* in the components *Main Motor* and *MiniServo* is used (see Figs. 7 and 8). Implemented on an FPGA the measurements have a deterministic resolution ($0.64e^{-6}s$ for MiniServo and $0.51e^{-6}s$ for Main). We define good performance as a deadline of two cycles of the 3kHz control loop, i.e. a latency $< 667\mu s$. For all applications, the host for the experimental setup is a Intel Core2Duo E6750 2.66GHz running QNX 6.3.

A. The Arm

The execution sequence of the inner loop of the control cascade, the main motor's control loop, is depicted by Fig. 9. The deterministic implementation yields a constant latency of exactly one clock cycle, i.e $10\mu s$.

Fig. 10 depicts the execution sequence of the outer host control loop. Two experiments were conducted to measure the latency of the host control loop. First, the latency is measured for a zero-computing dummy control application. Second, the latency is measured for the damping control algorithm presented in [25]. Table II lists the mean and standard deviation of the measured latencies for each of the four main motors. For the dummy application (HAL), the latency stays below the cycle time of $333\mu s$. On average, the damping control case exceeds the cycle time. Typically, this leads to more collisions which result in a higher jitter. This is reflected by the higher standard deviation. The histogram plots in Fig. 11 illustrate the higher jitter for the damping case (blue bars).

Application	Shoulder1	Shoulder2	Upp. Arm	Elbow
HAL mean [μs]	219.34	227.65	237.23	245.99
std [μs]	16.53	16.53	15.00	15.01
Damp. mean [μs]	343.24	363.09	382.50	393.48
std [μs]	39.34	41.50	37.21	34.10

TABLE II

THE LATENCY OF THE ARM MAIN MOTOR CONTROL LOOPS

Application	Grp 1	Grp 2	Grp 3	Grp 4	mean
Imped. mean [μs]	276.1	287.0	294.7	302.7	290.1
std [μs]	12.1	19.0	14.6	21.1	16.7

TABLE III

THE LATENCY OF THE HAND MINISERVOS CONTROL LOOP

B. The Hand

As for the arm, the overall latency of the control application loop was measured for the finger impedance control application presented in [1]. Fig. 12 (left) depicts the histogram of the measured latencies for the impedance control loop for all MiniServos. The plot of the mean latencies for each motor (Fig. 12 right) shows four distinct groups. These coincides with the four groups aggregated by the implementation on the FPGA (see Sec.V). Fig. 13 depicts the histograms for each group. Table III lists the mean latency and standard deviation for each group. The latency stays well below the deadline of $667\mu s$. On average, it even stays below one cycle.

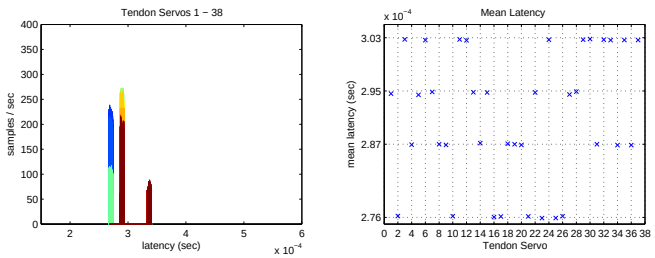

Fig. 12. Latency measured of the host impedance controller for all 38 hand MiniServos. Left: The histogram normalized to samples per second. Right: The mean latencies fall in four groups

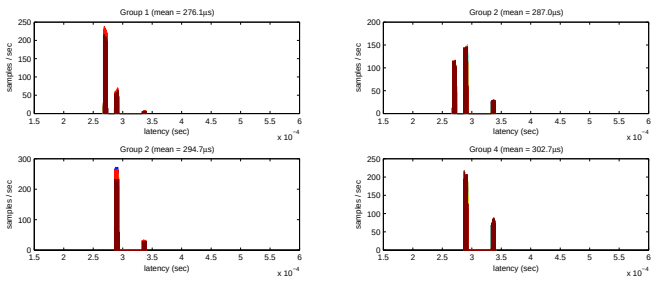

Fig. 13. The latency histograms for the four servo groups. These groups coincide with the aggregated servos of Fig. 8.

VII. Conclusion and Outlook

First control application implementations for the DLR Hand Arm System's *Computing and Communication Architecture* show that high performance control together with a flexible architecture, a high level hardware abstraction, and the domain model *Virtual Path* result in a deterministic and performant system with a convenient interface.

The experiments demonstrate that the desired deadline of two cycles ($667\mu s$) are met by the architecture.

However, the focus of those first control experiments is on the evaluation of the novel variable stiffness mechanisms. While the computing and communication architecture has proven to work well for this, we will conduct more experiments to further prove the robustness and stability of the architecture.

The four-layer hierarchy has resulted in a design of the electronic hardware that fits well into a highly-integrated mechatronic humanoid arm of the same size as a human arm. One exception is the design of the Arm Node (C3/M1 of Fig. 3). The combination of module and composition layer on one board breaks the hierarchy. This turned out to be impractical because the one-to-one relation of power inverter and computing node reduces scalability, power efficiency and maintainability. However, this experience is further confirmation that the hierarchical approach is suitable.

The next step will be to further reduce the latency by using the architecture's flexibility for the optimization of the communication implementation. This involves the movement of packet routing and signal aggregation from the host down to the joint FPGAs.

Future work will include the implementation of higher-level control algorithms of the arm's main motor on the FPGA in order to gain a 100kHz cycle.

Moreover, a framework for the automatic generation of the entire signal component infrastructure from application specifications will be developed to enhance the middleware approach.

The resulting flexibility of the architecture will be used to experiment with highly distributed algorithms, such as low-level safety measures or autonomous reflex actions.

References

[1] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, H. Hppner, S. Jörg, M. Nickl, A. Nothhelfer, F. Petit, J. Reill, N. Seitz, T. Wimböck, S. Wolf, T. Wüsthoff, and G. Hirzinger, "The DLR hand arm system," in *Proc. IEEE International Conf. on Robotics and Automation*, Shanghai, April 2011.

[2] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi, "Humanoid robot HRP-2," in *In IEEE Int. Conf. Rob. Aut*, 2004, pp. 1083–1090.

[3] Y. Ogura, H. Aikawa, H. Kondo, A. Morishima, H. ok Lim, and A. Takanishi, "Development of a new humanoid robot wabian-2," in *in Proceedings of IEEE International Conference on Robotics and Automation*, 2006, pp. 76–81.

[4] F. Pfeiffer, K. Löffler, M. Gienger, and H. Ulbrich, "Sensor and control aspects of biped robot "johnnie"," *I. J. Humanoid Robotics*, vol. 1, no. 3, pp. 481–496, 2004.

[5] J.-Y. Kim, I.-W. Park, J. Lee, M.-S. Kim, B. kyu Cho, and J.-H. Oh, "System design and dynamic walking of humanoid robot KHR-2," in *Proc. IEEE International Conf. on Robotics and Automation*, April 2005, pp. 1431–1436.

[6] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi, "Humanoid robot HRP-3," in *Proc. IEEE International Conf. on Robotics and Automation*, Nice, France, September 2008, pp. 2471–2478.

[7] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: System overview and integration," in *Proc. International Conference on Intelligent Robots and Systems*, 2002, pp. 2478–2483.

[8] W. Garage, *PR2 user manual*. [Online]. Available: http://pr2support.willowgarage.com/wiki/PR2_Manual

[9] T. Asfour, K. Regenstein, J. S. P. Azad, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An integrated humanoid platform for sensory-motor control," in *6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 169–175.

[10] S. Lohmeier, T. Buschmann, and H. Ulbrich, "System design and control of anthropomorphic walking robot LOLA," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 6, pp. 658 –666, Dec. 2009.

[11] Ch. Ott, O. Eiberger, W. Friedl, B. Bäuml, U. Hillenbrand, Ch. Borst, A. Albu-Schäffer, B. Brunner, H. Hirschmüller, S. Kielhöfer, R. Konietschke, M. Suppa, T. Wimböck, F. Zacharias, and G. Hirzinger, "A humanoid two-arm system for dexterous manipulation," in *Humanoids*, Genoa, December 2006.

[12] M. Diftler, J. Mehling, M. Abdallah, N. Radford, L. Bridgwater, A. Sanders, S. Askew, D. Linn, J. Yamokoski, F. Permenter, B. Hargrave, R. Platt, R. Savely, and R. Ambrose, "Robonaut 2 - the first humanoid robot in space," in *Proc. IEEE International Conf. on Robotics and Automation*, Shanghai, China, May 2011.

[13] B. Bäuml and G. Hirzinger, "Agile robot development (aRD): A pragmatic approach to robotic software," in *Proc. International Conference on Intelligent Robots and Systems*, Peking, October 2006.

[14] Data distribution service portal. OMG. [Online]. Available: http://portals.omg.org/dds/

[15] The robot application programming interface delegate project. NASA. [Online]. Available: http://robotapi.sourceforge.net/index.html

[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[17] The modular controller architecture. Http://www.mca2.org/. [Online]. Available: http://www.mca2.org/

[18] H. Hirohisa, K. Fumio, and K. Shuuji, "OpenHRP: Open architecture humanoid robotics platform," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, R. Jarvis and A. Zelinsky, Eds. Springer, 2003, vol. 6, pp. 99–112.

[19] *ECSS E-50-12A SpaceWire - Links, nodes, routers and networks*, European Cooperation for Space Standardization (ECSS), http://spacewire.esa.int, 2003.

[20] *BiSS Interface Protocol Description (C-Mode)*, C1 ed., IC Haus, http://www.ichaus.com, 2007.

[21] J. Liu, J. Eker, J. W. Janneck, X. Liu, and E. A. Lee, "Actor-oriented control system design: A responsible framework perspective," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 250–262, 2004.

[22] S. Jörg, M. Nickl, and G. Hirzinger, "Flexible signal-oriented hardware abstraction for rapid prototyping of robotic systems," in *Proc. International Conference on Intelligent Robots and Systems*, Peking, October 2006, pp. 3755 – 3760.

[23] A. Benveniste and G. Berry, "The synchronous approach to reactive and real-time systems," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1270 – 1282, September 2001.

[24] M. Nickl, S. Jörg, and G. Hirzinger, "The virtual path: The domain model for the design of the MIRO surgical robotic system," in *9th International IFAC Symposium on Robot Control*, IFAC. Gifu, Japan: http://www.ifac-papersonline.net/, 2009, pp. 97–103.

[25] F. Petit and A. Albu-Schäffer, "State feedback damping control for a multi dof variable stiffness robot arm," in *Proc. IEEE International Conf. on Robotics and Automation*, Shanghai, China, 2011.

[26] S. Wolf, O. Eiberger, and G. Hirzinger, "The DLR FSJ: Energy based design of variable stiffness joints," in *Proc. of the IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011.