Versatile Modular Electronics for Rapid Design and Development of Humanoid Robotic Subsystems

Brennand Pierce and Gordon Cheng
Institute for Cognitive Systems,
Technische Universität München,
Karlstrasse 45/II, 80333 Munich, Germany

***Abstract*— The development of humanoid robots is a highly complex process. Consisting of numerous subsystems which need to be combined for actuation such as electric motors or hydraulic valves, as well as sophisticated sensors and sensing units like skin. In this paper, we introduce our strategy to simplify the development of these complex electrical systems by using modular components. We took a generalised approach in the realisation of our design. By utilising a standard module based around a Field Programmable Gate Array (FPGA) it will be shown that our design is effective in terms of flexibility as well as scalability while at the same time ensuring compatibility over multiple generations and types of humanoid robotic systems. In this paper, we will present a few examples of humanoid sub-systems showing the benefits of this approach.**

I. Introduction

In comparison to other robots, the development of humanoid robotic systems creates unique electrical challenges. First of all, the weight and limited space for the electronics need to be considered. Secondly, the large variety of sensors which need to be interfaced have to be taken into account. Lastly, and more importantly, the demand for hard real time, high frequency control loops for the predictable control of this complex system has to be met. Over the past decades, extensive humanoid systems have been built [1], [2], [3], [4], [5]. Most of the designers for these humanoids have had a large tendency to re-implement their electrical systems in order to keep up with technological advancements and to interface with any new sensors and actuators. This is due to a lack of commercially available systems that fit with the evolution of the development cycle of humanoid robots. Thus, a path that can keep up with the advancing technologies is surely needed.

A. Motivations

Unlike other well-developed robotic systems, like wheeled robots where the controllers and sensors can easily be taken off-the-shelf and then integrated, developers of humanoid systems place a higher priority on compactness. This reduces the overall system's size and weight as much as possible. To reduce the amount of electrical cables running from the actuators and sensors to a centralised controller, developers have taken a distributed approach [2], [6], [5], where the low level controllers are distributed around the humanoid. This leads to the benefit of reduced cables as well as compactness. However, it does pose its own problem: How to handle with minimal latency the vast quantity of information that needs to be communicated to the central processing unit, while also ensuring a fully operational system at the same time?

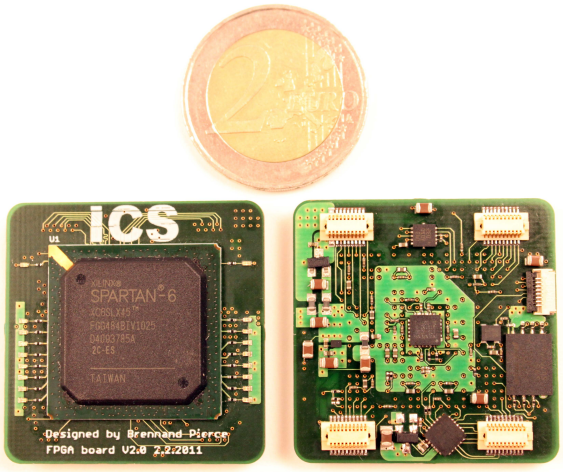

Fig. 1. Version 2.0 of the FPGA Module.

As these distributed controllers can be placed in close proximity to the sensors and actuators they control, they have the advantage of minimal latency between them and the low level control loop implemented on the controller. This leads us to a list of requirements for a compact humanoid electrical system:

- Distributed controllers capable of interfacing with all sensors and actuators on a single limb;
- Compact electrical design for minimising size and weight;
- Capabilility of high frequency, hard real time control loops;
- High speed, low latency intercommunication between limbs;
- Clear development path.

B. Development path – accounting for the future

This leads to the selection of the embedded processor and its complimentary specialist components, while also taking future developments into account. On the one hand, we need to choose an architecture that can essentially allow us to "plan ahead" for future changes in the humanoid hardware.

On the other hand, we also need to accommodate the increasing processing load of humanoids. In considering pros and cons of different DSP, FPGA, PIC and ARM processors in various combinations, we have chosen a stand alone FPGA. Unlike other systems that use an FPGA in conjunction with a DSP [4], we make use of *SoftCores*, a MIPS CPU instantiated on the FPGA's fabric for the high-speed closed loop control. At the moment, we have a single 80Mhz *SoftCore*, but we have the option to instantiate multiple *SoftCores* on a single FPGA if required.

The main advantage of using an FPGA is its parallel nature, which has demonstrated that complex algorithms can be optimized to outperform their DSP competitors [7]. Developers can also add new sensors, actuators and equations to the controller without interfering with the realtime nature of the already working system with each new element having their own independent loop frequency. The FPGA can be reconfigured with new features and I/O standards without redesigning it. For example unlike DSPs, the number of PWM, I2C or RS232 ports can be changed without having to redesign the circuit. The fact that I/O pins can be easily reconfigured to interface with any digital signal from the sensors without additional hardware has been seen as a further benefit. Although the utilisation of FPGAs are not new to humanoids [2], [4], [8], the technology has matured to a point where they now have greater advantages over older systems and DSPs. The only disadvantage is the knowledge required to implement algorithms on FPGAs. But with the increasing FPGA programming community directly implementing algorithms onto the device, tools simplifying this threshold are becoming available. For example, Handel-C allows developers to write parallel code in C, then translates the code directly to VHDL. The complexity of using FPGAs for algorithms will decrease in future as more feature rich tools are released. Thus, we will be able to increase the complexity of algorithms implemented without the need for *SoftCores*. It has been shown that a modular FPGA approach can reduce the development time and make the overall development easier [9], [10].

II. System Overview

At the heart of our system is a distributed network of FPGA modulars. In our design, every limb on the humanoid robot has a single FPGA module and these modules are networked together in a star formation. This module is then integrated into a custom carrier board that is tailored for each limb which all have their own sensors and actuation requirements. This star network has a "Central Controller" in the centre, which is a combination of FPGA and duo core ARM processors. These ARM processors are running FreeRTOS which is a realtime OS on one core, that takes care of the low level control and Ubuntu on the second core. There is an ethernet connection from the central controller to the High-Level PCs where all the cognitive processing is computed. In Fig. 2, an overview of this system is presented.

The advantages of having an FPGA module over integrating the FPGA straight onto the carrier board:

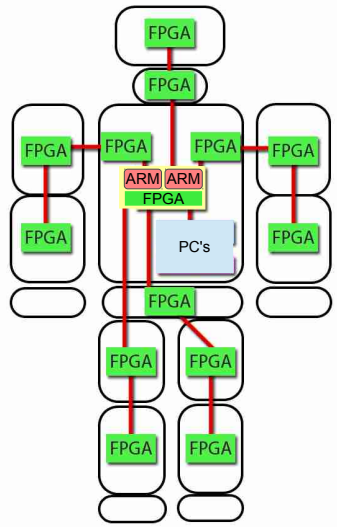

Fig. 2. Overall system design. The green FPGA squares are the modules. The red lines are bi-directional communication channels. The yellow hub is the central controller.

- The debugging of the electronics needs only be done once for a single FPGA module;
- The modular design also benefits from lower manufacturing costs as a single modular design can be ordered in large quantity, whereas carrier boards tend to be lower in volume;
- We can easily incorporate electronic components into each module that each limb will need, for example ADCs, gyroscopes or accelerometers directly soldered onto the module;
- It reduces development time;
- Simple daughter boards can be made to test new components or motor drivers without having the complexity of redeveloping the FPGA circuity;
- The modularising of the FPGA allows us to upgrade the complete system when a new FPGA comes to the market without having to redesign all the carrier boards again;
- Stacking the module makes it higher and but also means it has a smaller footprint;

A. FPGA Module

Fig. 1 shows the current version of the FPGA module which is based around an Xilinx Sparten 6 FPGA. It also has integrated onto the module common components that are desired on each limb, for example ADC and IMU. It currently consists of:

- Xilinx Spartan-6 LX45 FPGA.
- 6 axis IMU, MPU-6050.
- 8 Channel ADC, ADS8332.
- 2 x Full-duplex 200 Mb/s interconnection.
- 128Mb SPI Flash.
- 70 I/O, User configurable.
- 17 programable LEDs
- Size: 35mm x 35mm.

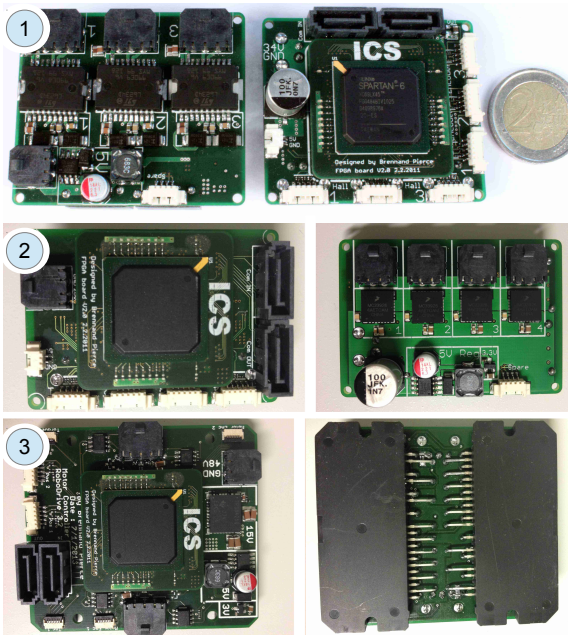

Fig. 3. Example of are Control boards. 1) low amp BLDC motor controller. 2) Brushed motor controller. 3) high amp BLDC motor controller.

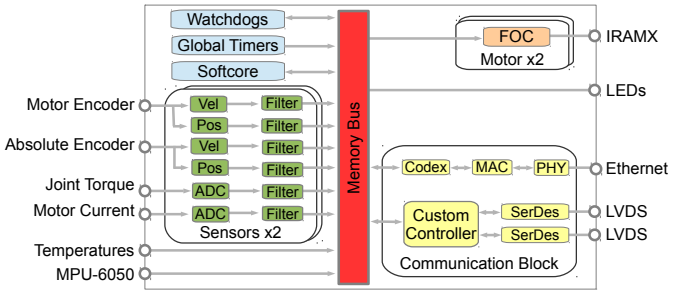

Fig. 4. Overview of the FPGA code structure for the "High Current BLDC Controller".

TABLE I

EXAMPLE OF DIFFERENT CARRIER BOARDS.

Carrier board	High Amp	Low Amp	Brushed
Brushless Motor	2x 20A/48V	3x 5A/36V	0
Brushed Motor	0	0	4x 5A/28V
16bit encoder	2	3	4
Ethernet base 10T	1	1	1
Fast Interconnectors	2	2	2
Torque sensor	2	0	0
Quadrature encoder	2	0	0
Size (mm)	68 x 60	56 x 48	60 x 42
Weight (g)	60	26	20

- Weight: 8g.

In order to develop our FPGA module, we simulated the power requirement and the optimal placement of the de-coupling capacitor network and the rating of the power regulating circuit for a fully configured FPGA. This shows the benefit of modularity, as we only had to simulate and design the circuit once. Due to the nature of the FPGA package (FGG 484) we had to use a six layered PCB board with 0.1mm wires to route the FPGA. These requirements make the PCB board expensive to manufacture. Where as from our experience designing carrier boards, those high specifications are not required and thus making the PCB boards for the carrier much cheaper. This is a big benefit as these simpler carrier boards tend to be a lot bigger then the FPGA board, thus saving on the overall system manufacturing costs. When the module is fully configured it only draws 40mA.

B. Control Boards

As it is foreseeable for any humanoid robot, each limb has its own electrical requirements due to different actuators and sensors. Thus each limb has a custom control board, examples of these can be seen in Fig. 3, and their specifications can be found in table I. These are the three example boards in more detail:

1) Low Amp BLDC controller: This board can be seen in Fig. 3.1 and was designed for "Mask-Bot 2i" [11], a robotic head. This board was designed to control a 3 DoF neck, which had a brushless motor which communicated via hall effect sensors and a digital encoder in each joint.

2) Brushed Motor Driver: This board can be seen in Fig. 3.2 This was developed for controlling a pair of eyes with 2 DoF each. The design is centred around 4 brushed motor controllers and the encoders. This is an example of a very simple 2 layer board.

3) High Current BLDC Controller: This board can be seen in Fig. 3.3 This board was developed for the thigh, which has high power requirements. It also had to interface with the frameless motors that have their own magnetic encoders. They also needed to interface with the foil strange gauges used to measure the torque at the joint. In Fig. 4 you can see an overview of the FPGA code structure. As you can see there are 4 main elements. The communication, sensor interface, BLDC motor controller and the *SoftCore* for the controller. All of these code blocks are joined together with shared memory and run in parallel.

The sensor section of this block diagram shows the advantage of the parallel approach, as each sensor can be sampled at its optimal frequency. For example the temperature sensor can only be sampled at 50hz whereas the ADC is sampled at 320khz. What this means is that we can over sample the ADC and pass it through a butterworth filter. Once all the sensors are processed we place them in the shared memory so the communication block and the *SoftCore* can access the data. From our experience with DSPs it is very hard to have this setup without a complex interrupt implementation.

C. Central Controller

At the centre of the system is the central hard realtime controller based on a Xilinx Zynq-XC7Z020, which is an FPGA and dual core ARM "A9 MPCore" processor running at 667MHz on a single chip. This controller has 512MB of DDR3 RAM, Gigabit ethernet, USB and an SD card reader. The FPGA fabric is used primarily for the communication to

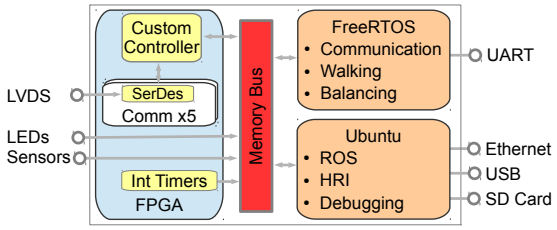

Fig. 5. Overview of the hardware structure of the central hub. This is based on an Xilinx Zynq-XC7Z020 with two ARM process.

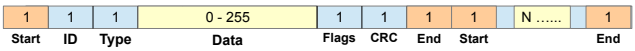

Fig. 6. The data structure of the High Speed Communication frame. Each sub-frame is separated with a start and stop unique word using 8/10b encoding.

the other FPGA modules, e.g acting like an intelligent hub. The central controller has 6 high speed communication ports as well as interfaces for a range of sensors.

The ARM processors is configured in an asymmetric multi-processing (AMP) configuration, with two separate OS running on each core. The first core is used for the hard realtime control loop. This loop is used for scheduling the network communication as well as the low level control that requires a fast update rate and also hard real time, for example the walking and balancing controller. This core runs RealRTOS, an open source real time operating system. The second core runs UBUNTU and Robot Operating System (ROS), which are used for the soft real time code, for example the path planning or human robot interaction. It also has a Gigabit ethernet connector to communicate to the High-Level PCs. The two cores and the FPGA fabric communicate via shared memory.

D. Communication

Communication is one of the main requirements for the electronics and is used to get data to and from the FPGA modules. Each module has two ways to communicate. The first is a simple ethernet controller based on 10Mb/s 10BASE-T communication and UDP packets, which is primarily used for debugging and to use the boards in a stand alone situation. The second is a high speed intercommunication used to daisy chain the boards together and to connect them to the central controller. The advantage of using an FPGA and 10BASE-T ethernet is that there is no need for any Physical Hardware Layer (PHY). As the FPGA is connected directly to the ethernet cable using 3.3V low voltage differential signal (LVDS), it saves costs and board space. The ethernet controller is implemented on the FPGA that handles the ARP requests and also IP and MAC address information.

1) High speed communication: For high speed intercommunication among FPGA modules, different standards were examined, including *EtherCAT* and *Powerlink*. Both are real-time protocols based on the Ethernet standard. [12] provide a good comparison of these two. They concluded that *EtherCAT* is a better standard for robotic systems. However, our evaluation shows that *EtherCAT* has some limitation for our requirements. The first is the speed; at the moment *EtherCAT* is only rated at 100Mb/s. The second is the physical electronic complexity. *EtherCAT's* protocol is based on sending Ethernet *telegrams*, this means the modules would require 2 $\times$ PHY Ethernet chips for simultaneous up and down stream traffics. Therefore, we investigated the possibility of using an FPGA on its own for the intercommunication. This produced a simple solution of direct pin to pin connection using a LVDS and 8/10b encoding, which has a maximum data rate of 1.05Gb/s. Modern FPGAs also have dedicated integrated multi Gb/s transceivers, with the current top of the range FPGA being capable of 28.05Gb/s, which shows there is an obvious design path for future communication bandwidth increase.

The communication network uses the FPGA fabric of the central controller as the master and all the FPGA modules are the slaves. Each module has 2 Rx and Tx interfaces, via these data are transferred bi-directional in full duplex configuration. The key to our protocol is that each module starts to forward the packet and appends packets on-the-fly. What this means is after the packet is received it is put into a FIFO and buffered for 4 bytes, then it is transmitted onto the next module in the network. The complete data structure we call a frame and consists of multiply sub-frames. All the sub-frames are transmitted one after the other and are separated by the reserved words *start* and *end* of sub-frame. Each sub-frame is targeted at a single module and consists of a unique ID for each module. When the module finds a sub-frame with its ID, it will read the data into RAM and at the same time will start to place its corresponding data onto the sub-frame. In this way each module on the network only adds a 4 byte delay to the overall latency for the complete network. The frame protocol is shown in Fig. 6. The data type tells the FPGA what the data structure is and the correct data length corresponding to this data type. The flag is used to tell the controller if the sub-frame was processed as well as giving error messages. The sub-frames are only processed on the downstream. The network will forward all upstream subframes without any buffering, thus the network can be configured with only a single cable running down each limb without the need for a return cable.

III. Discussions and Experimental Results

In this section 3 test setups are presented that highlight the key components of a humanoid robot system. 1) First is the humanoid test platform which is used to verify the *High Current BLDC Controller* electronics, which shows the FPGA can be used effectively for high frequency control of multiply high powered joints; 2) The second is a stereo vision system, where the FPGA controls 6 piezo motors simultaneously; 3) The last setup is for the development and verification of the high speed communication network.

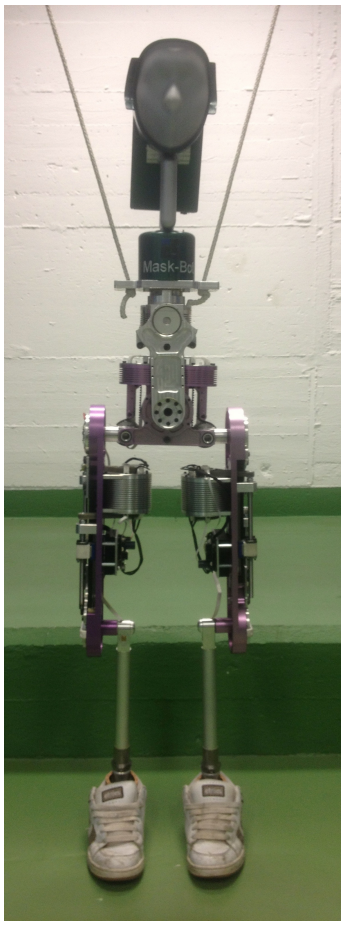

Fig. 7. The humanoid test platform. With 6 *High Current BLDC Controllers* in the main body and a single *Low Amp BLDC controller* for the head.

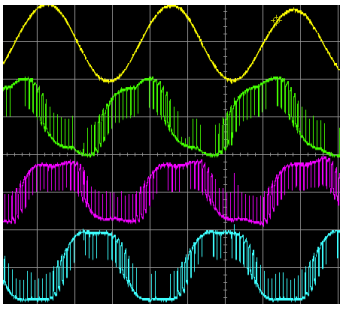

Fig. 8. Screen shot of SVPWM result taken with an oscilloscope. The yellow line is the current. Green, purple and blue are the voltage on each motor coil.

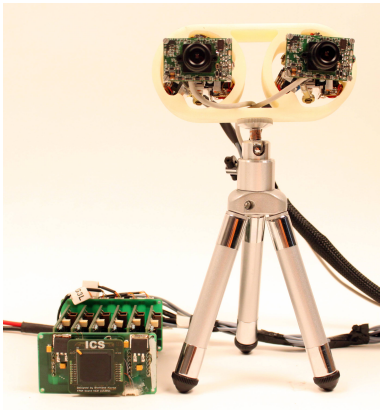

Fig. 9. High speed vision tracking system. This setup is used to demonstrate the FPGA can control 6 piezo motor in parallel. On the left is the electronics, where the FPGA is interfaced to 6 OEM piezo motor control boards and interfacing of 6 incremental encoders.

In developing this system we went through a number of electronic development cycles. This showed to us the advantage of separating the FPGA and the control boards. For example, when we updated the FPGA from a Spartan 3 to a Spartan 6, we only had to redesign the FPGA modular board once and then roll it out over the complete system.

A. *High Performance motor control*

As discuss earlier, it is critical to achieve high frequency control for humanoid robots, especially in this case of high performance controllers for permanent magnet synchronous motors (PMSM). Fig. 7 shows a torque controlled humanoid robot test platform. This was used to verify the performance of the *High Current BLDC Controller* from section II-B.3 which controls two high power PMSM motors from Robo-Drive [13] which are rated at 580 Watts each. For this robot we designed a complex feedback controller. The main control loop runs at 5khz while at the same time communicating with the central controller at 1khz. To implement the control algorithm we instantiated a *SoftCore* running at 5Khz. This gave the advantage to researchers that did not know hardware description language (HDL) but could still experiment with different controllers written in C. Once the controller has computed the desired current, the low level motor controller needs to produce the correct waveform for the 6 mosfets. This is achieved by using space vector modulation (SVM) running at 20 Khz and a magnetic encoder attached to the motor shaft as position feedback. The resulting voltage and current captured by an oscilloscope can be seen in Fig. 8 where the yellow plot is the current of a single coil and the green, purple and blue are the voltage on each motor coil. It can be seen that the current is a smooth sinusoidal waveform.

B. *Six DOFs Stereo Vision System*

Fig. 9 shows a high speed stereo vision system with 6 degrees of freedom (DoF): 2 $\times$ pan; 2 $\times$ tilt; 2 $\times$ torsion. The full mechanical structures and design of these eyes are presented in [14]. Each eye is controlled by 3 piezo motors which are used to control their position and velocity. These eyes show the benefit of a standard FPGA module, as it can be used to integrate commercial electronics into the system easily without having to change the communication structure or design of new electronics to integrate them into a master controller. In this system we used 6 “Fast, Compact OEM Ultrasonic Linear Motor” by Physik Instrumente (PI) GmbH & Co. KG [15] to drive the piezo motors and incremental encoders for position and velocity feedback.

This FPGA module receives desired angles alpha; beta and

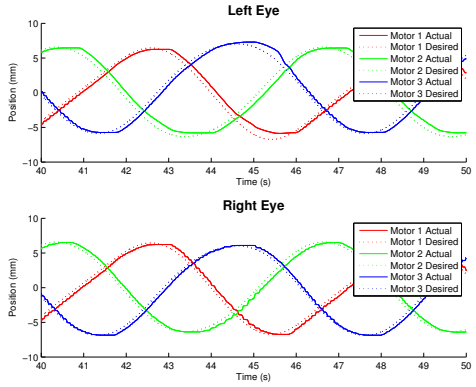

Fig. 10. Plot of 6 motors being controlled by one FPGA.

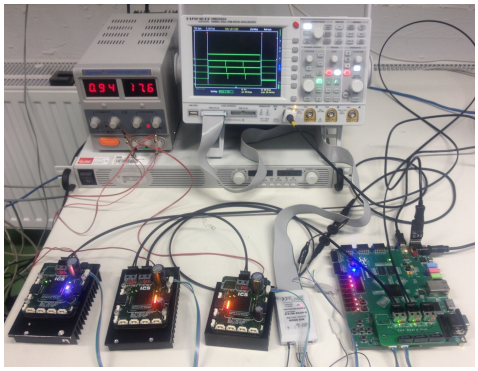

Fig. 11. This setup is for testing the network protocol and latency of the system with 3 modules and the central controller.

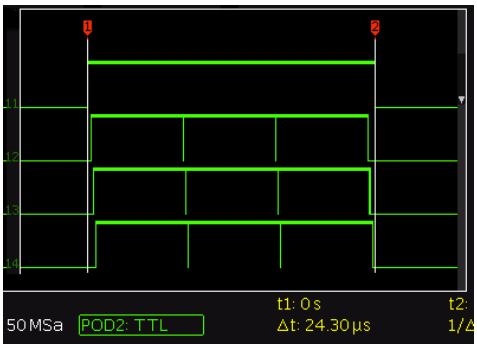

Fig. 12. This is a screenshot from the logic analyzer of the complete frame passing through the system. The top plot is the hub, set high when it receives a send signal and pulled low when the frame has been received and processed. The complete time for 3 modules each receiving 150 bytes of data takes 24.3μs.

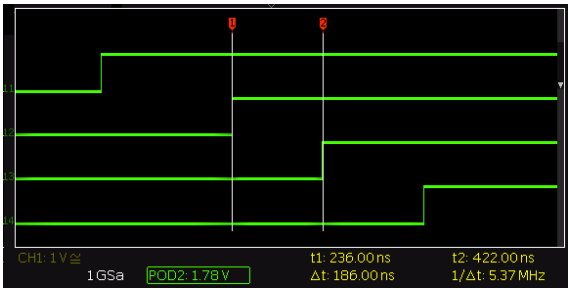

Fig. 13. This is the latency between FPGA modules. We set an IO pin high when the FPGA received a start of frame and captured using an logic analyzer. The latency between a module is 0.186μs.

gamma for each eye at 1khz from the central controller. The angles are then converted into desired motor positions and velocity using a *SoftCore* at 1khz. The desired position and velocity are then sent to 6 PID controllers which are running in parallel to generate the PWM signal for the motor drivers. This PID loop is running at 24.4khz. Fig. 10 shows the result of the eyes tracking desired position. This demonstrates that simultaneous high speed control of multiple DOFs is possible with our FPGA modules.

C. Network

Fig. 11 shows a setup for testing and debugging the *High speed communication* from section II-D.1. In this experiment, we send frames from the central controller to FPGA modules. The frame travels around the network until it is returned to the central controller. The central controller then checks the data processed and that it does not have any errors, as well as validates the returned data. This tests the integrity of the network. Using this setup we can also test the latency of the complete system using a logic analyzer and some of the FPGA's spare IO pins. On the central controller we set a debug pin high when the software sends a send command and pulls it low when the interconnect has received the complete frame on the RX line. This is used to measure the complete time a frame takes from the moment the software decides to send a frame until it is back and ready to be processed by the software. On the modules, we set a debug pin high when we receive a *start of frame* and low when we get an *end of frame*. This is shown in Fig.12. This shows the time it takes to send a frame from the central controller around the 3 modules with 150 bytes to each module and back. You can clearly see from this figure the 3 subframes and the very small latency between the modules. We repeated this experiment for different sized payloads and number of modules with the results displayed in Table II, where each module has two columns for measured and computed time for a complete frame.

To get a clearer idea of the latency between modules we can take a closer look at the beginning of the transmission which can be seen in Fig. 13. It shows a very predictable delay of 0.186μs between each module t_δ. You can also see that the time from the send signal to the first module receiving a frame t_s is 0.27μs This was also constant between all the experiments. The last measurement we took was the time between the last module in the network until the central controller had received and processed the signal t_r, this was 0.21μs and meant that the software could process the return frame. As the network is running at 200Mb/s and we are 8b/10b encoding the transmission we can also work out the time it takes to transmit the signal t_b which is 0.05μs. From all this we can come up with an equation that models this system to make predictions about the complete time for

TABLE II

MEASURED VS COMPUTED FRAME TIME IN μS.

	Number of modules					
	1		2		3	
Payload	Mea.	Comp.	Mea.	Comp.	Mea.	Comp.
50 bytes	3.18	3.42	6.24	6.35	9.32	9.28
150 bytes	8.20	8.42	16.24	16.35	24.30	24.28
250 bytes	13.24	13.41	26.30	26.35	39.30	39.28

TABLE III

THE TIME IN μS FOR A SINGLE TELEGRAM TO TRAVEL AROUND THE NETWORK AND RETURN TO THE CENTRAL CONTROLLER.

	Number of modules			
Payload	1	10	25	50
0 bytes	0.96	5.32	12.58	24.68
16 bytes	1.76	13.32	32.58	64.68
64 bytes	4.16	36.84	92.58	184.68
128 bytes	7.36	69.32	172.58	344.68
256 bytes	13.76	133.32	332.58	644.68

the system with different numbers of modules and payload lengths, this can be seen in equation 1.

$$t = t_s + t_r + \sum_{i}^{n} (M_i + 6)\, t_b + t_\delta \qquad (1)$$

where, t is the total time, t_s represents the send time, t_r is the receive time. M_i is the module payload for each $i = 1, 2, ..., n$ module, t_b is the time to transmit 1byte, 8b/10b encoded and finally, t_δ is the module latency which is constant.

To verify this equation we took a number of time measurements with the test setup with varying number of modules and payloads. These results can be seen in Table II and you can see that the measured results prove that our equation is correct. So we can make some real world prediction about the timing of the complete system. Table III provides a tabular of these timing results. We can also work out the latency for the complete humanoid with 13 modules each with a payload of 150 Bytes, this would be 104.27μs. This means we could communicate with the modules at 5Khz and still have time for the low level controller on the centre controller.

IV. CONCLUSION

Our FPGA module has been shown to fulfil the requirements to support the construction of humanoid robots. In this paper we showed that: 1) high demanding control can be accomplished for the main humanoid limbs; 2) multiple DoF can be controlled in parallel with a single FPGA module; 3) we can achieve high speed, low latency intercommunications for a network of modules.

We have also shown the advantage of a modular system by only having to design the FPGA once and then using it on different boards. This was highlighted when we upgraded from Spartan 3 to Spartan 6 where we only had to redesign the FPGA module once, then roll it out over our complete system so each limb could take advantage of the new feature, like added gate count.

As an example of the development path we are looking into upgrading the FPGA to Xilinx's new Artix 7. It has 30 percent lower power, 50 higher gate counts and is cheaper. It also has integrated ADC converters, thus decreasing the overall module price and complexity.

Over the years, through the development of the humanoid test platform, the advantage of the flexibility of the FPGA is highlighted. As it was easy to swap electronic components and we only had to reconfigure the FPGA to communicate with the new device. For instance, with the old FPGA, a Spartan 3an, we were only able to have a single joint working due to the limited resources. Upon upgrading to a Spartan 6, we now can have 2 complex joints operating and additional processing for other tasks.

ACKNOWLEDGMENT

This work was supported by the DFG cluster of excellence 'Cognition for Technical systems – CoTeSys' of Germany.

REFERENCES

[1] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The Development of Honda Humanoid Robot," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 1998, pp. 1321–1326.

[2] A. Nagakubo, Y. Kuniyoshi, and G. Cheng, "Development of a High-Performance Upper-Body Humanoid System," *Advanced Robotics*, vol. 17, no. 2, pp. 149–164, 2003.

[3] N. Kanehira, T. Kawasaki, S. Ohta, T. Isozumi, T. Kawada, F. Kanehiro, S. Kajita, and K. Kaneko, "Design and Experiments of Advanced Leg Module (HRP-2L) for Humanoid Robot (HRP-2) Development," *Magnesium*, vol. 2, no. October, pp. 2455–2460, 2002.

[4] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, *ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control*. IEEE, Dec. 2006.

[5] G. Cheng, S. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "CB: A Humanoid Research Platform for Exploring NeuroScience," *Advanced Robotics*, vol. 21, no. 10, pp. 1097–1114, 2007.

[6] F. Kanehiro, Y. Ishiwata, H. Saito, K. Akachi, G. Miyamori, T. Isozumi, K. Kaneko, and H. Hirukawa, *Distributed Control System of Humanoid Robots based on Real-time Ethernet*. IEEE, Oct. 2006.

[7] Z. Wen-Hong, "FPGA-based adaptive friction compensation for precision control of harmonic drivers," *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4657–4662, 2010.

[8] N. Ito, J. Urata, Y. Nakanishi, K. Okada, and M. Inaba, *Development of very small high output motor driver for realizing forceful musculoskeletal humanoids*. IEEE, Dec. 2010.

[9] R. Hartenstein, M. Servít, A. Dollas, B. Ward, and J. Babcock, "FPGA based low cost Generic Reusable Module for the rapid prototyping of subsystems - Field-Programmable Logic Architectures, Synthesis and Applications - Lecture Notes in Computer Science," pp. 259–270–270, 1994.

[10] S. Falsig and A. Soerensen, "An fpga based approach to increased flexibility, modularity and integration of low level control in robotics research," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 6119 –6124.

[11] P. Brennand, K. Takaaki, V. Christian, and C. Gordon, "Mask-Bot 2i: An active customisable Robotic Head with Interchangeable Face," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids2012)*, 2012, pp. 520–525.

[12] E. Baglini, G. Cannata, and F. Mastrogiovanni, *Design of an embedded networking infrastructure for whole-body tactile sensing in humanoid robots*. IEEE, Dec. 2010.

[13] RoboDrive GmbH. [Online]. Available: http://www.robodrive.de/

[14] T. Villgrattner and H. Ulbrich, *Optimization and dynamic simulation of a parallel three degree-of-freedom camera orientation system*. IEEE, Oct. 2010.

[15] Physik Instrumente (PI) GmbH & Co. KG. [Online]. Available: http://www.physikinstrumente.com/