2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Basic Implementation of FPGA-GPU Dual SoC Hybrid Architecture for Low-Latency Multi-DOF Robot Motion Control

Yuya Nagamatsu[1,2], Fumihito Sugai[1], Kei Okada[1], and Masayuki Inaba[1]

***Abstract*— This paper describes basic implementation of an embedded controller board based on a hybrid architecture equipped with an Intel FPGA SoC and an NVIDIA GPU SoC. Embedded distributed network involving motor-drivers or other embedded boards is constructed with low-latency optical transmission link. The central controller for high-level motion planning is connected via Gigabit Ethernet. The controller board with the hybrid architecture provides lower-latency feedback control performance. Computing performance of the FPGA SoC, the GPU SoC, and the central controller is evaluated by computation time of matrix multiplication. Then, the total feedback latency is estimated to show the performance of the hybrid architecture.**

I. Introduction

Robots with multi-Degrees-Of-Freedom (multi-DOF), including humanoid robots, multi-legged robots, and musculoskeletal robots, perform whole-body motion with their body parts interacting dynamically with environment and with each other. High-performance and low-latency control is required for multibody dynamics of robots. For example, our high-power humanoid robot JAXON [1] has 33 body joints and 39 motors driving them. For another example, our musculoskeletal humanoid robot Kengoro [2] has 174 DOF and 116 motors. Control computation becomes more complex as the DOF of the robot increases.

Feedback latency is the most important factor which determines the performance of online feedback control with some designated mathematical law. Feedback latency means the time lag from mechanical disturbance input to mechanical actuation output reflecting the input. The target factor in this paper is elements in feedback latency caused by electronic circuit and control computation. Note that properly handled feedback latency can sometimes provide interesting discovery. Ott [3] analyzed the control performance of the humanoid robot with configured feedback latency from the viewpoint of biology.

Control period often comes on the table discussing the performance of real-time robot control. Control period is one of the important elements determining feedback latency. Many robotics researchers set the target value of control period for dynamic robot motion to around 1 ms [4][5][6][7].

Actual control system for multi-DOF robots usually takes multilayer structure regarding both hardware and software. The total feedback latency accumulates while the feedback path goes through a number of layers. Thus not only the control period for each layer but also the number of layers and the data transfer latency between the layers are important factors. Refer to [8] for well summarized information about the data communication protocols in the recent humanoid robots.

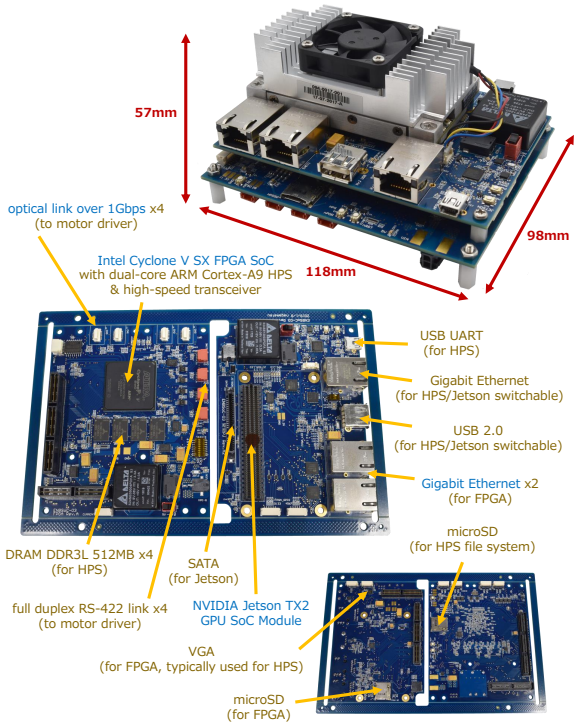


Fig. 1. Overview of the ENBSoC-03 controller board with FPGA-GPU dual SoC hybrid architecture

The embedded controllers can have lower data transfer latency compared to the central controller such as PC, because they are closer to the endpoints such as the motor-drivers. If the embedded controllers perform the fast whole-body feedback control which has been performed in the central controller so far, then the feedback latency will decrease and the control performance will be improved.

In recent years, acceleration technology for embedded controllers, including Field Programmable Gate Array (FPGA) and Graphics Processing Unit (GPU), is growing. Several robots are powered by acceleration with FPGAs [9][10] or GPUs [11][12].

We develop an embedded controller implementing hybrid architecture with an FPGA-based System-on-a-Chip (SoC) and a GPU-based SoC. An FPGA, a GPU, and corresponding CPUs coexist heterogeneously in the high-performance

[1]Authors are with the Graduate School of Information Science and Technology, the University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan
[2]nagamatsu@jsk.imi.i.u-tokyo.ac.jp

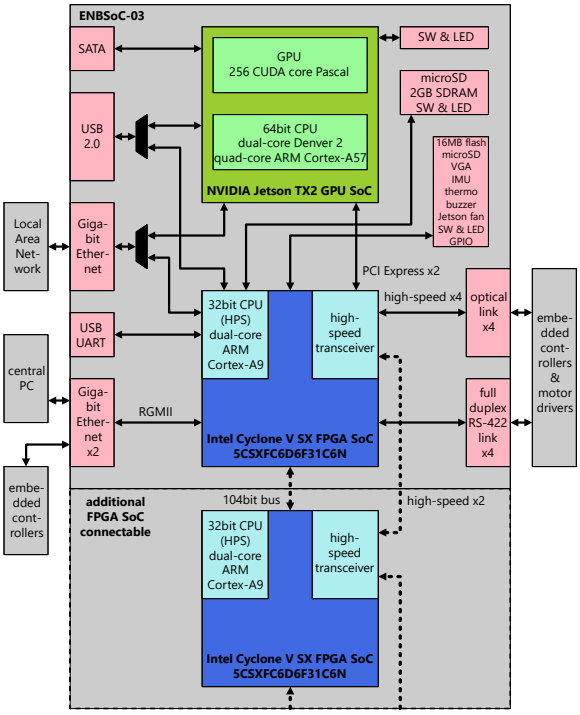


Fig. 2. Hardware architecture of the ENBSoC-03 controller board

embedded computing system. The controller is connected to the FPGA-based embedded network via high-speed optical transmission link, and connected to the central PC via Gigabit Ethernet. Our purpose is to enable the embedded controller to perform low-latency multi-DOF robot motion control, including dynamics computation, balance stabilizer, and model identification.

This paper introduces design and development of the controller board with the hybrid architecture. Then the basic latency performance of the controller is evaluated. Section II explains the design idea of the architecture and the controller. Section III evaluates the computing performance of the FPGA SoC and the GPU SoC compared to a typical PC. Section IV analyzes the total feedback latency including data transfer latency. Finally, we conclude this paper in Section V.

II. Design and Hardware Implementation

A. Hardware Overview

Fig.1 shows the ENBSoC-03 controller board introduced in this paper. Fig.2 shows the hardware architecture of the ENBSoC-03 controller board.

Intel Cyclone V SX FPGA SoC 5CSXFC6D6F31C6N and NVIDIA Jetson TX2 (Tegra X2 SoC module) are employed on the controller. Cyclone V SX is equipped with ARM Cortex-A9 CPUs and high-speed transceivers in addition to an FPGA. Jetson TX2 is equipped with Denver 2 and ARM Cortex-A57 CPUs in addition to a GPU.

Remaining part in this section describes the development history and the design idea of the FPGA-GPU hybrid architecture and the ENBSoC-03 controller board.

B. Development History

Our development genealogy strongly related to this work began with the local network for the high-power robots [13]. The central PC was connected to the FPGA-based embedded network via PCI bus. DMA-based PCI data transfer does not have so narrow band. However, PCI edge connector is too unstable to ensure vibratory dynamic motions of the high-power robots. PCI has some other practical disadvantages. The custom PCI board requires maintenance of the designated device driver as the PC or the Linux kernel version is upgraded. Recently PCI has been deprecated and incompatible with latest PCs.

Next, we developed the EtherCAT-based interface board [14][15] to eliminate the practical disadvantages of PCI. EtherCAT can be implemented on standard Ethernet interface, which is a de facto standard in recent PCs. The solid hardware connection of Ethernet has almost resolved the instability under impact and vibration which was a fatal problem with the PCI system. Several latest robots also employs EtherCAT interface [7][8][16]. However, EtherCAT has been based on 100 Mbps slow-speed Ethernet so far. In addition, packet-based EtherCAT transfer has non-negligible overhead time. These disadvantages give a bottleneck in improving feedback latency. Actually the 1 ms control period has been unreasonable while data size to transfer per period has increased for our new control schemes such as torque control [14]. Note that next EtherCAT G/10G is being standardized recently, so expectations rise for future.

At the almost same time, we developed the experimental low-latency controller with an FPGA SoC [17]. Joint stiffness control and dynamics computation by 200-400 μs period was implemented on the FPGA SoC. High-speed optical transmission was employed for embedded network. This work has drastically improved feedback latency of relatively simple control. However, only simplified dynamics computation can be implemented with only the FPGA SoC.

Now, we introduce the GPU SoC in addition to the FPGA SoC in order to implement more complex low-latency dynamic robot motion control. We also introduce Gigabit Ethernet interface, which improves data transfer latency between central PC while retaining the advantages of EtherCAT interface. Brand new small package design puts these technologies to practical use in the robot body with restricted space for components.

C. Architecture Design

Custom logic circuit on the Cyclone V SX FPGA SoC conducts the low-latency data transfer in the ENBSoC-03 controller board.

Optical transmission link constructs low-latency embedded network between motor-drivers or other distributed controllers. Optical transmission has advantages not only in the large data throughput but also in the robust ability to transfer data under electromagnetic field noise. Optical fiber can also decrease the volume of the cable compared to electric transmission. Unfortunately, not every distributed controllers can implement optical transmission due to size and cost of the

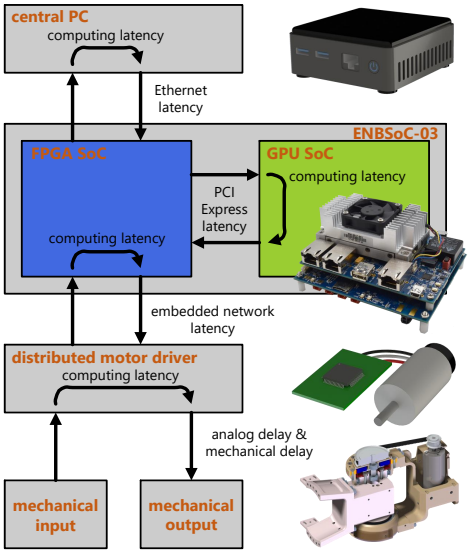


Fig. 3. Feedback latency elements in hierarchical control system

transceiver. Thus conventional RS-422 electric differential link is also employed.

Dual Gigabit Ethernet interface with Microchip Technology PHY KSZ9031RNX is utilized to communicate with the central PC or additional Ethernet controllers. The PHY is controlled via RGMII interface. We have been often dissatisfied with ready-made controllers such as EtherCAT modules because of their behavioral specification, including transfer rate and buffer size. The raw RGMII interface enables us to develop some custom controller from scratch. The Socket API of the OS can reduce maintenance cost of the software in the central PC.

The Jetson TX2 GPU SoC module accesses the FPGA SoC via PCI Express x2 interface. Custom PCI Express device can be implemented in the FPGA. The GPU SoC can also exchange data directly with external controllers via Ethernet.

The CPU in the FPGA SoC, Hard Processor System (HPS), is connected to FPGA via direct synchronous bus, while the HPS has less computing power than the GPU SoC. Thus we can construct complementary control system, in which the FPGA SoC takes charge of simple process with hard latency requirement and the GPU SoC handles complex control. This is the core feature of the hybrid architecture. The central PC also has a complementary relationship between the ENBSoC-03 controller board.

Additionally attachable FPGA SoC submodules can provide scalability for the ENBSoC-03 controller board system. FPGA can perform very low-latency control as long as the resource usage allows. Neighbor FPGAs are connected via 2 channels of high-speed transceivers and 104 bit general purpose logic bus.

III. Analysis of Computing Performance

Fig.3 shows elements of total feedback latency. Total feedback latency is composed of mechanical delay, data transfer latency between modules, and control computation time in every module. This section evaluates computing performance of FPGA SoC, GPU SoC, and central PC.

Inverse dynamics computing, calculating joint torque for desired robot motion, is a typical instance of computation required in recent force-based robot control. Recursive Newton-Euler method [18] is well known to be efficient with complexity linear to DOF of the robot. Definite complexity of Newton-Euler method depends on the robot model and implementation. Around dozens of 3-to-6-order matrix or vector operations are required per joint number.

We evaluate the computing performance of each module by implementing abstract matrix operations. Iterative multiplication of 6-order square matrices with 32 bit integer elements is implemented here. Algorithm 1 shows an outline of the test code. This is equivalent to calculating $\boldsymbol{B} = \boldsymbol{A}^{\mathrm{ITERATION}+1}$. Integer matrix multiplication overflows soon and the correct upper bits of the result cannot be obtained, but this makes no problem for computing performance evaluation. Floating point operations make little difference from integer operations, except for the case on the FPGA logic circuit.

Algorithm 1 Test Code with Matrix Multiplication

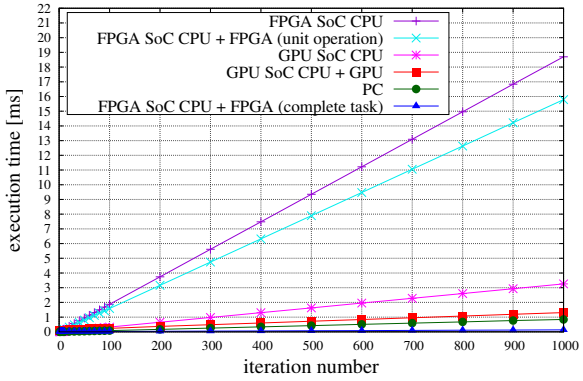
```
ORDER = 6, ITERATION = 1 to 1000
provide A ∈ ℕ^(ORDER×ORDER) (ℕ = uint32)
B = A
for i = 1 to ITERATION do
  C = AB
  B = C
end for
```

We compare the FPGA SoC, the GPU SoC, and a typical laptop PC, Lenovo T460s with Intel Core i7-6600U CPU, by executing Algorithm 1 . Fig.4 shows the average result of 10000 trials for every condition.

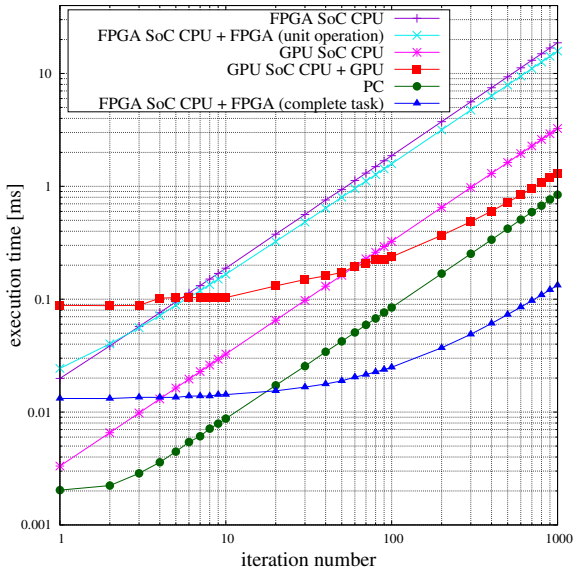
Ubuntu 18.04.4 LTS on Linux 4.15.0 operates the 925 MHz 32 bit dual-core ARM Cortex-A9 CPU on the FPGA SoC. Ubuntu 18.04.4 LTS on Linux 4.9.140 operates the 2035 MHz 64 bit dual-core Denver 2 and quad-core ARM Cortex-A57 CPU on the GPU SoC. The 256 CUDA core Pascal GPU on the GPU SoC is driven in 1135 MHz. Ubuntu 18.04.4 LTS on Linux 4.15.0-88-generic operates the 2600 MHz Intel Core i7-6600U CPU of the PC.

The ENBSoC-03 controller board is equipped with an onboard current sensor for each of the two power input lines. One powers the whole motherboard including the FPGA SoC, and the other powers the GPU SoC module. The power consumption in idle state, while executing the test code, and with around maximum CPU load is respectively calculated into Table I based on the current and the supply voltage. Both the FPGA SoC and the GPU SoC consume around 5 W. Therefore, comparison of the execution time also gives good reference for the power consumption.

"FPGA SoC CPU" in Fig.4 shows the result with simple iteration in the CPU on the FPGA SoC, in which single 6-order square matrix multiplication takes 6^3 times of integer multiplication and 6^3 times of integer addition. It takes



(a) Linear scale view



(b) Log scale view

Fig. 4. Execution time of iterative 6-order square matrix multiplication with FPGA SoC, GPU SoC, and PC (showing average of 10000 trials, "FPGA SoC CPU": Cortex-A9 CPU on FPGA SoC, "FPGA SoC CPU + FPGA (unit operation)": CPU on FPGA SoC with FPGA-implemented single-shot matrix-vector multiplication circuit, "GPU SoC CPU": Denver 2 & Cortex-A57 CPU on GPU SoC, "GPU SoC CPU + GPU": CUDA accelerated multiplication on GPU SoC, "PC": Core i7-6600U CPU on T460s laptop PC, "FPGA SoC CPU + FPGA (complete task)": CPU on FPGA SoC with FPGA-implemented iterative matrix multiplication circuit)

longer time than other comparative CPUs. In case of 1 ms control period, the computation limit will be less complex than 50 times of matrix multiplication.

"FPGA SoC CPU + FPGA (unit operation)" in Fig.4 shows the result with combinational single-shot matrix-vector multiplication circuit on the FPGA. The CPU accesses the multiplication circuit every time of unit matrix-vector multiplication. Matrix-vector multiplication circuit is employed because the FPGA does not have enough resources for 6-order matrix-matrix multiplication. This condition results in almost close time to "FPGA SoC CPU". The data transfer time between the CPU and the FPGA is not so less than the time required for multiplication on the CPU.

"FPGA SoC CPU + FPGA (complete task)" in Fig.4 shows the result with iterative matrix multiplication circuit as a complete task executor on the FPGA. This condition shows excellent performance. The CPU first provides the matrix $\boldsymbol{A}$ and the iteration number ITERATION for the FPGA, and receives the result when the FPGA notify calculation complete. The FPGA multiplies a matrix and a vector in a clock of 50 MHz, and finishes all operations in $6 \times \mathrm{ITERATION}$ clocks. In case of $\mathrm{ITERATION} = 1000$, computation in the FPGA takes just 120 μs. The total result is 133.1 μs including data transfer overhead. FPGA can provide some flexibly optimized accelerator specialized for the desired operation. There is a trade-off relationship between latency performance, versatility, and resource usage. For example, employing vector-vector multiplier instead of matrix-vector multiplier will result in 6 times latency and 1/6 times resource usage.

TABLE I
TYPICAL POWER CONSUMPTION OF EACH COMPUTER MODULE

Module	Idle	Test Code	Max CPU
FPGA SoC & Peripheral	4.95 W	5.10 W	5.25 W
GPU SoC Module	3.90 W	4.80 W	5.55 W
Intel Core i7 PC (Typ.)	(5-10 W)	-	(20-40 W)

"GPU SoC CPU" in Fig.4 shows the result of the CPU on the GPU SoC. This condition takes intermediate time between the CPU on the FPGA SoC and the PC shown in Fig.4 "PC".

"GPU SoC CPU + GPU" in Fig.4 shows the result with parallel computing on the GPU. The CPU first provides the matrix and wait the result, as with "FPGA SoC CPU + FPGA (complete task)". NVIDIA CUDA executes synchronous 6^2 parallel computing per matrix element. This condition shows comparable performance to the PC around 1000 times of iteration, while large CPU-GPU data transfer offset dominates in small numbers of iteration. The GPU SoC has a potential for complex and versatile robot motion control.

IV. ANALYSIS OF FEEDBACK LATENCY

A. Overview of Feedback Latency

This section analyzes the total feedback latency in Fig.3, considering data transfer latency between modules in addition to the result in Section III.

The feedback latency T_L caused by electronic circuit and control computation is determined with the computation time and the data size to transfer.

$$T_L = \sum_i T_C^{(i)} + \sum_j T_D^{(j)} \left(D^{(j)} \right) \tag{1}$$

$T_C^{(i)}$ is the computation time in the computing node i. $T_D^{(j)}$ is the time required for data transfer in the link j, and is a function of the data size $D^{(j)}$. Every latency of i and j accumulates into the total feedback latency T_L. $T_D^{(j)}$ can usually be approximated with a linear function with a positive offset.

$$T_D^{(j)} \left(D^{(j)} \right) = T_{DK}^{(j)} D^{(j)} + T_{DO}^{(j)} \tag{2}$$

The feedback latency in the simple model, in which the central controller receives sensor data D_U from the endpoint

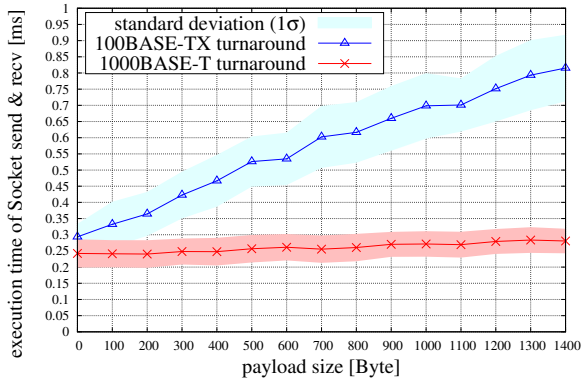


Fig. 5. Execution time of Ethernet packet turnaround with Linux PC and Socket API (showing average and standard deviation of 10000 trials)

and sends command data D_D to the endpoint, follows:

$$T_L = T_C + \left(\sum T_{DK}\right)(D_U + D_D) + 2\sum T_{DO} \quad (3)$$

These relationships will be helpful to determine the best module for feedback control to be implemented on, between the FPGA SoC, the GPU SoC, and the central PC. T_C tends to be large and $\sum T_D$ tends to be small in the order of the FPGA SoC, the GPU SoC, and the central PC.

B. Data Transfer Latency

Low-latency optical transmission link makes the embedded distributed network involving the FPGA SoC on the ENBSoC-03 controller board. The link is implemented with the high-speed transceivers and FPGA custom logic circuits. Thus the latency offset T_{DO} is very small. The transfer rate is variable. We have validated stable data transfer under 2500 MHz signal and 2000 Mbps data rate with 8b/10b encoding.

The central PC accesses the ENBSoC-03 controller board via Gigabit Ethernet. We evaluate the turnaround time of Ethernet with the T460s laptop PC and the ENBSoC-03 controller board. After the FPGA SoC receives a packet from the PC, it returns the packet, with its header partially modified, to the PC by store-and-forward. This simulates some turnaround protocol like EtherCAT. The PC measures the differential time from send command to recv command of the Linux Socket API. The Ethernet controller in the FPGA SoC is implemented in 2 different modes, 1000BASE-T (1000 Mbps) and 100BASE-TX (100 Mbps).

Fig.5 shows the evaluation result. There is offset time close to 300 μs in both transfer rates. 100 Mbps transfer causes around 800 μs time to transfer a 1400 Byte packet, so that 1 ms control period is about to fail. On the other hand, 1000 Mbps transfer stays around 300 μs for every packet size.

Standard Ethernet packet size is up 1500 Byte. When multiple packets are required, iteration of send + recv will make offset time in proportion to packet number. On the other hand, iteration of send and recv separated from each other will increase little offset time. In actual turnaround measurement of 5 packets with 1000 Byte payload, (send + recv)×5 takes 1195 μs, while (send×5 + recv×5) takes only 287 μs.

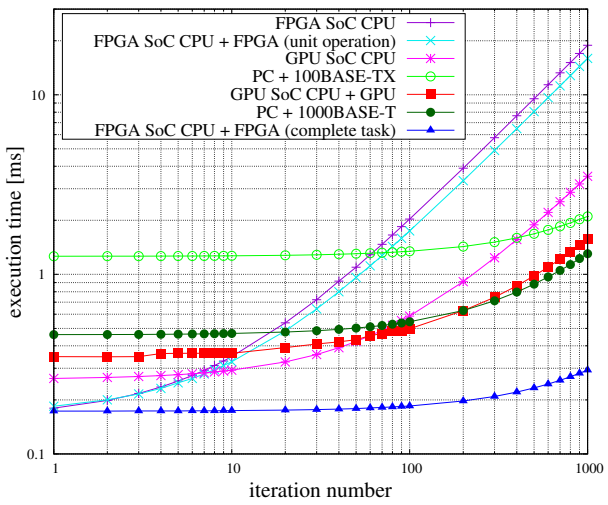


Fig. 6. Estimated total electronic feedback system latency with constant data transfer offset in addition to execution time of iterative 6-order square matrix multiplication in Fig.4 (FPGA SoC has 0.16 ms offset, GPU SoC has 0.26 ms offset, PC + 1000BASE-T has 0.46 ms offset, conventional PC + 100BASE-TX has 1.26 ms offset)

FPGA SoC and GPU SoC are connected via the onboard PCI Express Gen1 x2. The maximum transfer rate in the physical layer is 4000 Mbps. Overhead time for the CPU to access PCI Express [19] is smaller than Ethernet packet transfer overhead. However, overhead time close to 100 μs is still required like the CPU-GPU internal transfer time in Fig.4. A simple implementation of PCI Express system on the ENBSoC-03 board takes 4.3 μs turnaround time per word (up to 64 bit) by single-word memory access method, and it takes 0.4 μs slope per Byte by continuous memory transfer method. Therefore, in the example of 144 Byte matrix data, around 60 - 600 μs of transfer time is required depending on the data addressing design.

C. Estimation for Total Feedback Latency

Amount of data to be transferred via local network in some multi-DOF robot depends on the robot and the system configuration. Torque control experiments of our high-power humanoid robot [20] requires around 2000 Byte data turnaround per control period of 1 ms. Now, we estimate instance of total feedback latency.

2000 Byte data takes 8 μs for single optical transmission in 2000 Mbps. In case of turnaround in 10 hops network by store-and-forward on each node, total optical transmission latency accumulates into 160 μs. The central PC has additional latency at least around 300 μs for Gigabit Ethernet transfer. Conventional 100 Mbps Ethernet transfer requires around 800 μs for 2000 Byte payload in addition to Gigabit Ethernet. The GPU SoC has additional latency typically around 100 μs for exchanging data just on demand between the FPGA SoC via PCI Express, with little offset time for each transfer in comparison with Ethernet.

Fig.6 replots the data transfer latency offset in addition to Fig.4 (b).

The specialized accelerator circuit for iterative matrix multiplication on the FPGA takes the lowest latency. FPGA can provide some high-performance controller at the expense of resource usage. Interesting result appears excepting the FPGA-implemented iterative multiplier. To say with the complexity equivalent to 6-order square matrix multiplication, the FPGA SoC wins at no more than 7 times of iteration, otherwise the GPU SoC wins at no more than 200 times of iteration, and otherwise the PC wins. The performance of the accelerated GPU SoC still remains close to that of the PC above 200 times of iteration. Although whole-body feedback control with latency less than 1 ms had been difficult with 100Mbps Ethernet (i.e. EtherCAT), it became achievable with the ENBSoC-03 controller board. Note that this estimation does not include application software latency such as sleep time of periodic control task and overhead time for multilayer software.

V. Conclusion

Feedback latency has been a problem in dynamic motions of conventional multi-DOF robots, in which the central controller takes charge of allover whole-body control. We developed the high-performance embedded controller board equipped with FPGA-GPU dual SoC hybrid architecture and Gigabit Ethernet interface. It is shown that complementary low-latency feedback control system can be constructed involving the responsive FPGA SoC, the intermediate GPU SoC, and the high-performance central controller. Specialized in the designated function, the FPGA can also perform excellent low-latency feedback. Now, complex whole-body robot motion control with sub-millisecond scale latency is within our reach. Although the controller board is just a trial prototype now, we hope this work to provide helpful development infrastructure for new generation multi-DOF robots.

References

[1] Kunio Kojima, Tatsuhi Karasawa, Toyotaka Kozuki, Eisoku Kuroiwa, Sou Yukizaki, Satoshi Iwaishi, Tatsuya Ishikawa, Ryo Koyama, Shintaro Noda, Fumihito Sugai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Development of life-sized high-power humanoid robot jaxon for real-world use. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 838–843, November 2015.

[2] Yuki Asano, Kei Okada, and Masayuki Inaba. Design principles of a human mimetic humanoid: Humanoid platform to study human intelligence and internal body system. *Science Robotics*, Vol. 2, No. 13, p. eaaq0899, 2017.

[3] Christian Ott, Bernd Henze, Georg Hettich, Tim Niklas Seyde, Máximo A. Roa, Vittorio Lippi, and Thomas Mergner. Good posture, good balance: Comparison of bioinspired and model-based approaches for posture control of humanoid robots. *IEEE Robotics and Automation Magazine*, Vol. 23, No. 1, pp. 22–33, 2016.

[4] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludvic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, Vol. 40, No. 3, pp. 473–491, 2016.

[5] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Sourères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro. TALOS: A new humanoid research platform targeted for industrial applications. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 689–695, 2017.

[6] Tianyi Ko, Ko Yamamoto, Kazuya Murotani, and Yoshihiko Nakamura. Compliant biped locomotion of Hydra, an electro-hydrostatically driven humanoid. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 587–592, 2018.

[7] Kenji Kaneko, Hiroshi Kaminaga, Takeshi Sakaguchi, Shuuji Kajita, Mitsuharu Morisawa, Iori Kumagai, and Fumio Kanehiro. Humanoid robot HRP-5P: An electrically actuated humanoid robot with high-power and wide range joints. *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, pp. 1431–1438, 2019.

[8] Felix Sygulla, Robert Wittmann, Philipp Seiwald, Tobias Berninger, Arne-Christoph Hildebrandt, Daniel Wahrmann, and Daniel Rixen. An EtherCAT-based real-time control system architecture for humanoid robots. In *IEEE International Conference on Automation Science and Engineering*, pp. 483–490, 2018.

[9] Muhammad Yaqoob Javed, Yasir Saleem, Muhammad Majid Gulzar, Syed Tahir Hussain Rizvi, and Muhammad Saeed Tahir. Implementation of FPGA based efficient gait controller for a biped robot. In *International Conference on Robotics and Automation Engineering*, pp. 42–47, 2017.

[10] Hsuan-Ming Feng, Ching-Chang Wong, Chih-Cheng Liu, and Sheng-Ru Xiao. ROS-based humanoid robot pose control system design. In *IEEE-RAS International Conference on Systems, Man, and Cybernetics*, pp. 4089–4093, 2018.

[11] Benjamin Chrétien, Adrien Escande, and Abderrahmane Kheddar. GPU robot motion planning using semi-infinite nonlinear programming. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 10, pp. 2926–2939, 2016.

[12] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *IEEE International Conference on Robotics and Automation*, pp. 7087–7094, 2018.

[13] Junichi Urata, Yuto Nakanishi, Kei Okada, and Masayuki Inaba. Design of high torque and high speed leg module for high power humanoid. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4497–4502, 10 2010.

[14] Yuya Nagamatsu, Takuma Shirai, Hiroto Suzuki, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Distributed torque estimation toward low-latency variable stiffness control for gear-driven torque sensorless humanoid. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5239–5244, September 2017.

[15] Kento Kawaharazuka, Tasuku Makabe, Shogo Makino, Kei Tsuzuki, Yuya Nagamatsu, Yuki Asano, Takuma Shirai, Fumihito Sugai, Kei Okada, Koji Kawasaki, and Masayuki Inaba. TWIMP: Two-wheel inverted musculoskeletal pendulum as a learning control platform in the real world with environmental physical contact. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 784–790, November 2018.

[16] Tamim Asfour, Mirko Wächter, Lukas Kaul, Samuel Rader, Pascal Weiner, Simon Ottenhaus, Raphael Grimm, You Zhou, Markus Grotz, and Fabian Paus. ARMAR-6: A high-performance humanoid for human-robot collaboration in real-world scenarios. *IEEE Robotics and Automation Magazine*, Vol. 26, No. 4, pp. 108–121, 2019.

[17] Takuma Shirai, Yuya Nagamatsu, Hiroto Suzuki, Shunichi Nozawa, Kei Okada, and Masayuki Inaba. Design and evaluation of torque based bipedal walking control system that prevent fall over by impulsive disturbance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 739–746, October 2018.

[18] John M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 10, No. 11, pp. 730–736, 1980.

[19] Wei Wang, Miodrag Bolic, and Jonathan Parri. pvFPGA: Accessing an FPGA-based hardware accelerator in a paravirtualized environment. In *International Conference on Hardware/Software Codesign and System Synthesis*, pp. 1–9, 2013.

[20] Shintaro Komatsu, Yuya Nagamatsu, Tatsuya Ishikawa, Takuma Shirai, Kunio Kojima, Yohei Kakiuchi, Fumihito Sugai, Kei Okada, and Masayuki Inaba. Humanoid robot's force-based heavy manipulation tasks with torque-controlled arms and wrist force sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3055–3062, November 2019.