Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A VHDL application for kinematic equation solutions of multi-degree-of-freedom systems*

Hüseyin Oktay ERKOL†[1], Hüseyin DEMİREL[2]

([1]*Department of Mechatronics Engineering, Faculty of Technology, Karabük University, Karabük 78050, Turkey*)

([2]*Department of Electrical and Electronics Engineering, Faculty of Engineering, Karabük University, Karabük 78050, Turkey*)

†E-mail: oktayerkol@karabuk.edu.tr

Received Apr. 1, 2014; Revision accepted July 17, 2014; Crosschecked Nov. 13, 2014

Abstract: As kinematic calculations are complicated, it takes a long time and is difficult to get the desired accurate result with a single processor in real-time motion control of multi-degree-of-freedom (MDOF) systems. Another calculation unit is needed, especially with the increase in the degree of freedom. The main central processing unit (CPU) has additional loads because of numerous motion elements which move independently from each other and their closed-loop controls. The system designed is also complicated because there are many parts and cabling. This paper presents the design and implementation of a hardware that will provide solutions to these problems. It is realized using the Very High Speed Integrated Circuit Hardware Description Language (VHDL) and field-programmable gate array (FPGA). This hardware is designed for a six-legged robot and has been working with servo motors controlled via the serial port. The hardware on FPGA calculates the required joint angles for the feet positions received from the serial port and sends the calculated angels to the servo motors via the serial port. This hardware has a co-processor for the calculation of kinematic equations and can be used together with the equipment that would reduce the electromechanical mess. It is intended to be used as a tool which will accelerate the transition from design to application for robots.

Keywords: Multi-degree-of-freedom systems, Kinematics, Co-processor, Serial communication, Six-legged robot
doi:10.1631/jzus.C1400120 **Document code:** A **CLC number:** TN43; TP242

1 Introduction

Nowadays, servo motors controlled by common serial communication protocols have become widespread with technological developments. They have internal closed-loop control hardware. Motors controlled via the serial port can be used by connecting to a single data and power line. It makes designers' work easier in practice. These types of motors reduce the load on the main processor. This advantage is undeniable, especially in robotic applications which involve a large number of servo motors and a high degree of freedom.

However, kinematic equations are complicated because of the high degree of freedom in many multi-joint structures. Using a co-processor for calculations is the necessity when fast and smooth motions are asked for (Taira *et al*., 2005; Zheng *et al*., 2012; Zhu *et al*., 2013). The co-processor makes the necessary calculations by using parameters taken from the main processor. Co-processors are used to reduce the load on the main processor in applications that require high speed. They are designed to work in real time in most applications. Hani *et al*. (2006) developed a crypto processor for safety in their study. Barron-Zambrano *et al*. (2012) made a study using field-programmable gate array (FPGA) to provide real-time robot motion. Juang *et al*. (2013) worked on a co-processor which calculates the joint angles of a climbing robot. Zheng *et al*. (2012) studied kinematic calculations of a multi-degree-of-freedom (MDOF) robot and used FPGA

* Project (No. KBÜ-BAP-13/1-DR-011) supported by the Department of Bilimsel Araştırma Progeleri, Karabük University, Turkey
ORCID: Hüseyin Oktay ERKOL, http://orcid.org/0000-0002-3595-175X

and a coordinate rotation digital computer (CORDIC) algorithm.

Six-legged robots maintain their popularity today and are the subject of various studies. Mahapatra and Roy (2009) made a simulation study of the mechanical structure and the forces occurring in the legs of a six-legged robot. Shih *et al*. (2012) made a study of different walking algorithms. Sandoval-Castro *et al*. (2013) focused on the kinematic equations in their study. As for Pa and Wu (2012), they have worked on a general purpose robot.

We take a look at studies on serial communication: Idris *et al*. (2006) designed a serial port which has a built-in self-test and is applied on FPGA. Hardware design of a standard serial communication and simulation were performed in Very High Speed Integrated Circuit Hardware Description Language (VHDL) in a study conducted by Fang and Chen (2011). Again, an FPGA-based serial port interface was used for connection between digital input-output cards and central processing units by Fongjun *et al*. (2011).

In this study, a kinematic co-processor which calculates the required joint angles is designed. This co-processor has internal serial ports and can be used with motors that have an internal controller and serial ports. The designed hardware calculates the required joint angles and sends them via the serial port to a six-legged robot in which each leg has three joints. At the same time, it obtains the necessary parameters to calculate joint angles via the serial port from the main processor. The design is done in VHDL language and tested on an FPGA board.

2 Mechanical structure and kinematic equations

A general structure of the six-legged robot whose control is targeted in this study and the details of three-joint legs are given in Fig. 1.

The forward and inverse kinematics studies of Murray *et al.* (1994), Roennau *et al*. (2010), and Zhu *et al*. (2013) can be examined. The defined Denavit-Hartenberg parameters for the leg structure in Fig. 1 are given in Table 1.

Using these parameters, the forward kinematic equation of the structure can be obtained as given in

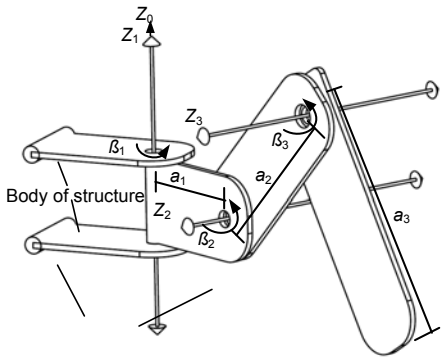

Fig. 1 Basic leg structure and joints of the six-legged robot
Z_0 axis is the reference axis, Z_1 axis is the normal of the robot body, and Z_2 and Z_3 axes are parallel to the robot body. The angle between Z_1 and Z_2 axes is 90°

Table 1 Denavit-Hartenberg parameters for three-joint legs

i	α_i (°)	a_i (mm)	d_i (mm)	β_i
1	0	0	0	β_1
2	90	60	0	β_2
3	0	80	0	β_3

i: joint index; α: the fixed angle between Z_i and Z_{i-1} axes; a: the length of each part (link) in Fig. 1; d: the coordinate of the part along z; β: the variable angle around the Z axis

Eq. (1) (Siciliano *et al*., 2009). It is called the transformation matrix and defines the position and orientation of the foot (end effectors). Herein, the notations c_i and s_i are the abbreviations for $\cos\beta_i$ and $\sin\beta_i$, respectively. The notations $s_{i,j}$, $c_{i,j}$ denote respectively $\sin(\beta_i+\beta_j)$, $\cos(\beta_i+\beta_j)$.

$$\boldsymbol{T}_3^0=\begin{bmatrix} c_{23}c_1 & -s_{23}c_1 & s_1 & c_1(a_2c_2+a_3c_{23}) \\ c_{23}s_1 & -s_{23}s_1 & -c_1 & s_1(a_2c_2+a_3c_{23}) \\ s_{23} & c_{23} & 0 & a_3s_{23}+a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

The last column of the matrix in Eq. (1) gives the position of the foot. Based on this, the x, y, and z coordinates of the foot are as follows:

$$P_x=(d_2+d_3)s_1+a_2c_1c_2+a_3c_1(c_2c_3-s_2s_3), \quad (2)$$

$$P_y=-(d_2+d_3)c_1+a_2s_1c_2+a_3s_1(c_2c_3-s_2s_3), \quad (3)$$

$$P_z=a_3s_{23}+a_2s_2. \quad (4)$$

We can obtain angle β_i (i=1, 2, 3) by solving Eqs. (2)–(4). When they are solved, s_i and c_i (i=1, 2, 3)

are obtained. After this step, β_i can be computed as follows:

$$\beta_N = \arctan\left(\frac{s_N}{c_N}\right). \quad (5)$$

The information will be sent to the servo motors for moving the foot to the desired point.

3 Serial port

A serial port is an electronic hardware and can transmit one data bit at any time. It is generally bidirectional so that information can travel in both directions at once. The most important advantages of serial ports are the small number of data lines and small-sized connectors (Axelson, 2007; Erkol and Demirel, 2013). The industry has a tendency to develop devices that can be controlled remotely. Today, motors and motor drivers controlled via a serial port have become widespread. The hardware developed in this study is designed to control these types of motors.

The data transfer rate is widely supported between 600 and 155200 bits/s by RS232. The frequently used form is 8-*N*-1, i.e., eight data bits and one stop bit without parity. An example serial line waveform in asynchronous mode is shown in Fig. 2. It is an 8-bit data packet. Line status is logic 1 in waiting position. The receiver gets ready to take data bits when the line jumps to logic 0. It reads all subsequent bits at the right time using the internal clock. If the transmission rate is 9600 bits/s, one bit time is 1/9600=0.0001 s. The internal clock starts to count when the receiver receives the start bit. The receiver reads the first bit after the first 0.0001 s, the second bit after the second 0.0001 s, etc. In this way, the receiver reads all data bits between the start and stop bits. Then it switches to standby until a second start bit. The transmitter sends the 8 bits in a similar way. It generates start, stop, and parity bits using the internal clock.

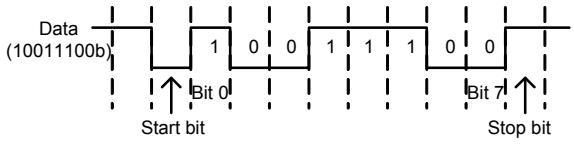

Fig. 2 Data transmission signal

3.1 Communication protocol

The hardware in this study is designed to work with motors controlled with a serial port. HerkuleX DRS-0101 digital servo motors (Dongbu Robot, Korea) are used as servo motors. The most important feature of this type of motor is the serial interface. These motors can be connected to a single data and power line. Up to 254 motors can be connected to a single line. Operation voltage is 7.5 V; the signal level of the data line is at transistor-transistor logic (TTL) levels, and the angular resolution is 0.325°. The rotor position range is 0–320º. It is developed for robotic applications. Pin descriptions of the motor are given in Table 2 and the general connection scheme is illustrated in Fig. 3.

Motors are controlled with commands by a serial line. An internal proportional-integral-derivative (PID) controller adjusts the rotor position using the received angle information. The communication format is 8-*N*-1. Communication speed can be adjusted between 0.0576 and 0.677 Mb/s. The length of the data packets is 7–223 bytes. The package structure is given in Table 3.

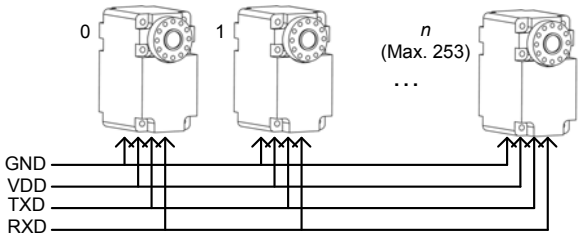

Fig. 3 Motor connections

Table 2 Pin descriptions of the motor connector

Pin index	Name	Description
1	GND	Electrical ground connection
2	VDD	Supply voltage
3	TXD	Transmitting line
4	RXD	Receiving line

Table 3 Details of the communication packet

Item	Value	Length (byte)	Item	Value	Length (byte)
Header	0xFF 0xFF	2	CheckSum1	–	1
Pack size	7–223	1	CheckSum2	–	1
pID	0–0xFE	1	Data [*n*]	–	Max. 216
CMD	1–9	1			

Header is standard and 2-byte long. Each motor has its own identity number called pID. It can be between 0 and 253. CMD is the actual command operated by the motor controller. There are nine different commands for Herkulex. CheckSum1 and CheckSum2 are used to control transmission errors and calculated as

$$\text{CheckSum1}=(\text{PacketSize}\wedge\text{pID}\wedge\text{CMD}\wedge\text{Data}[0]\wedge\text{Data}[1]\wedge\ldots\wedge\text{Data}[n])\&0\text{xFE}, \quad (6)$$

$$\text{CheckSum2}=\sim(\text{CheckSum1})\&0\text{xFE}, \quad (7)$$

where '^' is the bit exclusive OR operator, '&' is the AND operator, and '~' is the bit NOT operator. Data[i] ($0\leq i\leq n$) is the parameter of command and changes for every command. The packet structure to move the rotor to 90º is given in Table 4. For detailed information, the Herkulex manual can be examined.

4 Recommended structure of the chip

General blocks of the design are given in Fig. 4. First, commands and data are obtained from the receiving data pack. If the received command is a position command, kinematic equations are solved using the received parameters. Then the angles are sent to the motors by packaging, as described in Section 3.1. It does not require any calculation if the received command is not a position command. In this case, only the relevant data is sent to the motors.

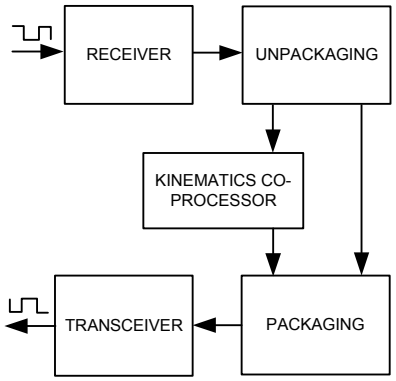

Fig. 4 Block diagram of design

4.1 Serial port hardware

Transmitter and receiver block diagrams of the designed system are shown in Fig. 5. All blocks of the hardware were performed using the VHDL language.

The 'baud rate generator' generates clock signals required for serial transmission. Start bit, stop bit, and 8-bit data are transferred to the data line using the generated clock signals. The process is controlled by an 'algorithmic state machine'. At the receiver end, the process made by the transmitter is reversed. The internal clock generates a clock signal on the same frequency as the transmitter. Using the internal clock, the 'algorithmic state machine' receives the bits and saves them to the data memory.

4.2 Packaging

The designed hardware uses two types of packets. The first type is used for commands that do not require parameters. The second type is used for commands that require parameters. There are three commands that do not require parameters, i.e., motor_on, motor_off, and clear_errors. These commands affect all motors in the system, because all motors are off/on when the robot is turned off/on. When a fault occurs, the solution to eliminate this error must be generated by the main processor that uses this

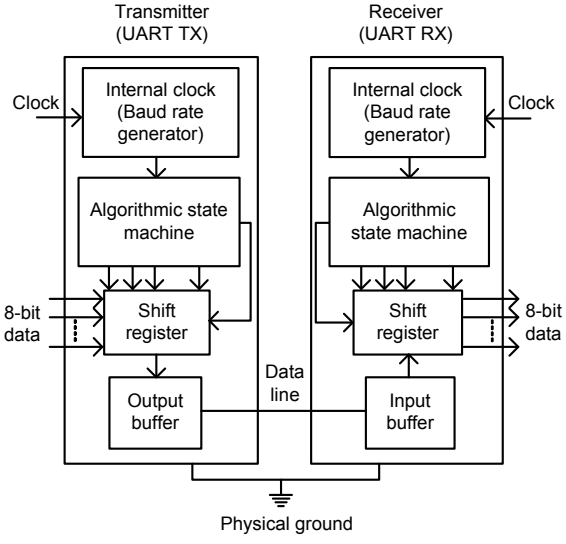

Fig. 5 Block diagram of the transmitter and receiver

Table 4 Data sequence to move the rotor to position of 90°

Header	Pack size	pID	CMD	CheckSum1	CheckSum2	Data[1]	Data[2]	Data[3]	Data[4]	pID
0xFF 0xFF	12	253	6	6	248	10	0	2	4	253

hardware. Error messages can be deleted using the 'clear_errors' command after the problem is solved. The commands used and the corresponding structures are given in Table 5. Packages are 3-byte long. The first two bytes together represent a header. The third byte is the command byte. In the designed system, all numbers greater than 36 are accepted as a command that does not require parameters. Numbers equal to or less than 36 are accepted as a position command and the number specifies the number of bytes.

Another command is the 'go_position' command. This command includes the position data. These data are used by the motor. The packet structure of the 'go_position' command is shown in Table 6 and its parameter structure is given in Table 7. If the first number after the header is 36, 30, 24, 18, 12, or 6, it is accepted as the 'go_position' command. At the same time, the 'go_position' command is equal to the byte number of position data. This structure is repeated for each leg. The variables in position parameters specify the position of the foot in Cartesian coordinates.

The general process block diagram is given in Fig. 6. The number after receiving the header is checked. There is no calculation if the number is 78_H, 82_H, or $8C_H$. In this case, related commands are sent to all motors and returned to the beginning. If the number is not greater than 36, kinematic calculations are performed using the parameters, and the angles are sent to the motors in a package.

Table 5 General command structure

	Header	Command	Data
Torque_on	0x41 0x41	78_H	–
Torque_off	0x41 0x41	82_H	–
Clear_errors	0x41 0x41	$8C_H$	–
Go_position	0x41 0x41	02_H	6–36 bytes

Table 6 Command structures requiring parameters

	Header	Command	Position parameter 1	Position parameter 2	Position parameter *N* (Max. 6)
Value	0x41 0x41	6–36	–	–	–
Length (byte)	2	1	6,12,18, 24,30,36	6,12,18, 24,30,36	6,12,18, 24,30,36

Table 7 Structure of position parameters

Symbol	$(+/-)_{Px}$	P_x	$(+/-)_{Py}$	P_y	$(+/-)_{Pz}$	P_z
Value	2B/2D	0–FF	2B/2D	0–FF	2B/2D	0–FF
Length (byte)	1	1	1	1	1	1
Function	Sign	Position	Sign	Position	Sign	Position

ASCII codes (2B/2D) refer to signs of numbers (+/−)

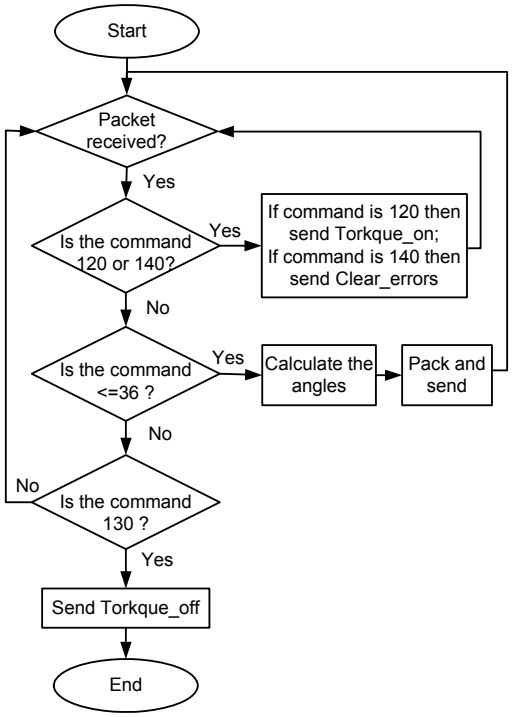

Fig. 6 Block diagram of the process

4.3 Kinematic co-processor

All data used in communications are declared as the type of an 8-bit vector. If received data is a number, it is subjected to several transformations before being used. There are P_X, P_Y, and P_Z in the packet when a 'go_position' command is received. At first, the position parameters are converted from vectors to fixed-point numbers. If the sign is negative, this number is multiplied by '−1'. The received numbers are measured in millimeter and they show the position of the foot. After receiving P_x, P_y, and P_z positions, joint angles are calculated by using the kinematic equations given in Section 2. Trigonometric functions were performed using lookup tables. Angle values are converted to a number range of 0−1023 after calculations, because motors accept rotor position information in this range. Rotor positions and related values are given in Fig. 7. Conversations are made using Eq. (8). The angle values are packaged as

shown in Table 6 after scaling by the conversations.

$$\text{Angle}_{0-1023} = \frac{(\alpha + 166.7) \times 1024}{333.4}. \quad (8)$$

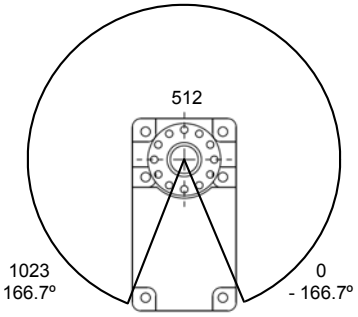

Fig. 7 Scanned rotor area

5 Simulation

The hardware designed using the VHDL language was simulated by ModelSim SE-64 10.1c. The given simulations were made at a speed of 115200 bits/s. This speed can be set to any desired value. It was tested at 9600, 57200, and 115200 bits/s, respectively. The main clock frequency of FPGA was 50 MHz. The signals used in the simulation and the corresponding interpretations are listed in Table 8.

5.1 Receiver simulation

The signals that occurred during the process of receiving data '01000001' are given in Fig. 8. Two stop bits and one start bit were used with each byte. All bits took the same time and it was 1/115200=8.68 μs for the speed of 115200 bits/s. When the first stop bit was received, the 'ONE BYTE IS RECEIVED' signal was generated. There were two stop bits on the signal. Receiving two stop bits took 2×8.68=17.36 μs, as shown in Fig. 8. Eight-bit data transmission including start and stop bits took 11×8.68=95.48 μs.

The signals that occurred during the process of receiving data (41 41 06 2B 64 2B 00 2D 64)$_H$ are shown in Fig. 9. The transfer of each byte with start and stop bits took 9×8.68+17.36=95.48 μs. 'ONE BYTE IS RECEIVED' signal was generated for each byte. 06$_H$ was received after header (41$_H$ 41$_H$). The number was a 'go_position' command and showed also the length of coming data in bytes. As soon as the command data was received, the 'COMMAND' signal obtained the value of 02$_H$ (go_position). 'PACK IS

Table 8 Signals used in the simulation and interpretations

Signal	Interpretation
SYSTEM CLOCK	Main clock signal
RX	The receiving signal from the data transmission line
TX	The generated signal on the transmission line
DATA	The received/sent 8-bit data
BIT NUMBER	The order of data bits
ONE BYTE IS RECEIVED	Generated when the 8-bit data is received
PACKAGE IS RECEIVED	Generated end of the package receiving process
ONE BYTE IS TRANSMITTED	Generated when 8-bit data transmission is completed
PACKAGE IS TRANSMITTING	Indicate that a package is being transmitted

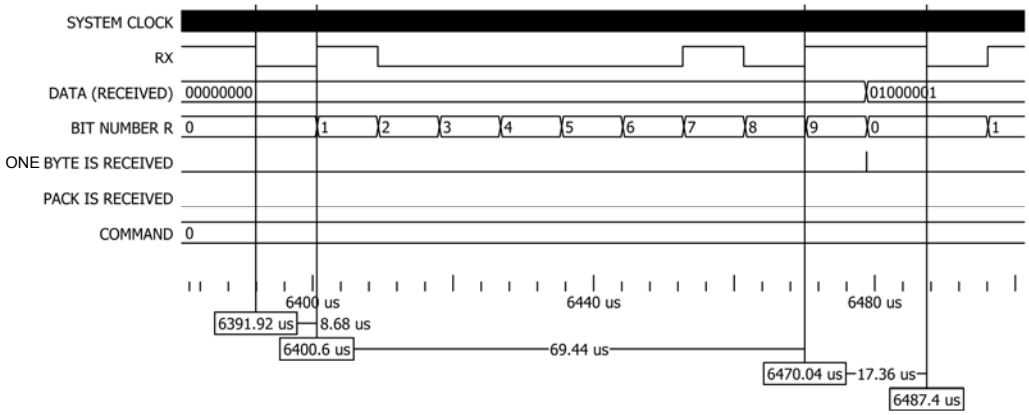

Fig. 8 Receiver simulation
'0' is the start bit, '9' is the stop bit, and '1–8' are data bits

RECEIVED' signal has been generated with the last received data. Thus, the packet receiving process was completed and the system was ready to receive the next packet. Next, kinematic calculations and the packaging process were performed.

5.2 Transmitter simulation

The signals that occurred during the process of transmitting data '00000110' are given in Fig. 10. Two stop bits and eight data bits were used. Parity bits are not used. At the same time, the stop bits referred to the idle position of the line. All bits took the same time and it was 1/115200=8.68 μs for a speed of 115200 bits/s. The transmission of 8-bit data with start and stop bits took a time of 11×8.68=95.48 μs. 'ONE BYTE IS TRANSMITTED' signal was produced after the second stop bit. Now the system was ready to send the next data.

Fig. 11 shows the transmission of generated packets after kinematic calculations. It started to send packages when the 'SEND CREATED PACKS' signal was generated. 'ONE BYTE IS TRANSMITTED SIGNAL' was generated after transmission of each byte. During the packet transmission process, the 'PACKAGE IS TRANSMITTING' signal was at logic 1. It became logic 0 at the end of the sending process.

5.3 Co-processor simulation

Fig. 12 shows the time lag between the receiving and transmitting processes. The calculation and packaging process were done sequentially after 'PACK IS RECEIVED' signal. After this process, the 'CALCULATION AND PACKAGING ARE COMPLETED' signal was generated. Package transmission started with 'SEND CREATED PACKS' signal. The time between calculation and the transmitting processes was 37.64 μs. Fig. 13 shows the details of 'CALCULATION AND PACKAGING ARE COMPLETED', 'SEND CREATED PACKS', and 'PACKAGE IS TRANSMITTING' signals.

6 Hardware test

Test inputs have been created with the Matlab/Simulink program as embedded for the hardware test.

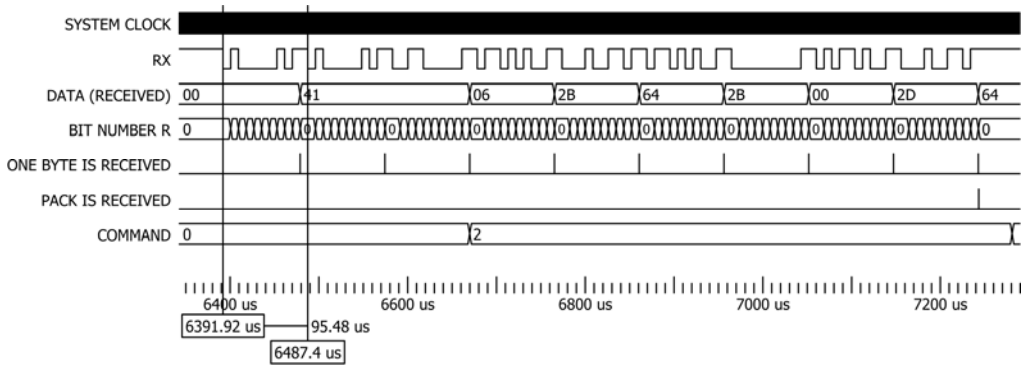

Fig. 9 Receiving a command packet

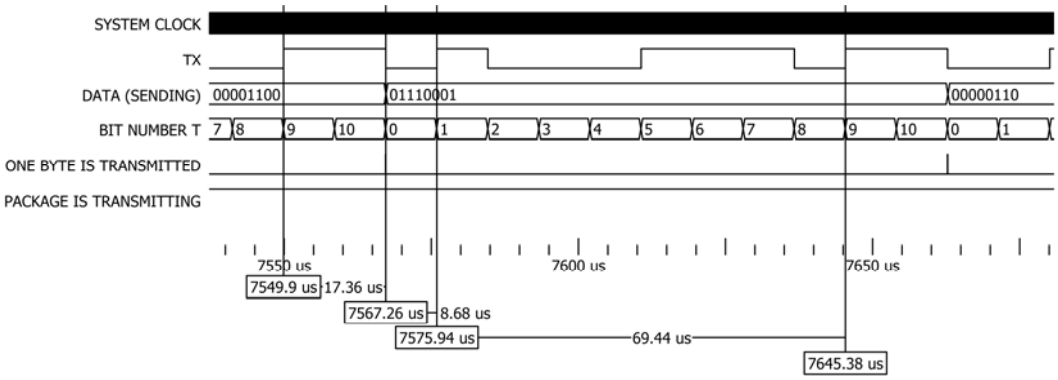

Fig. 10 Transmitter simulation

'0' is the start bit, '1–8' are data bits, and '9' and '10' are the stop bits

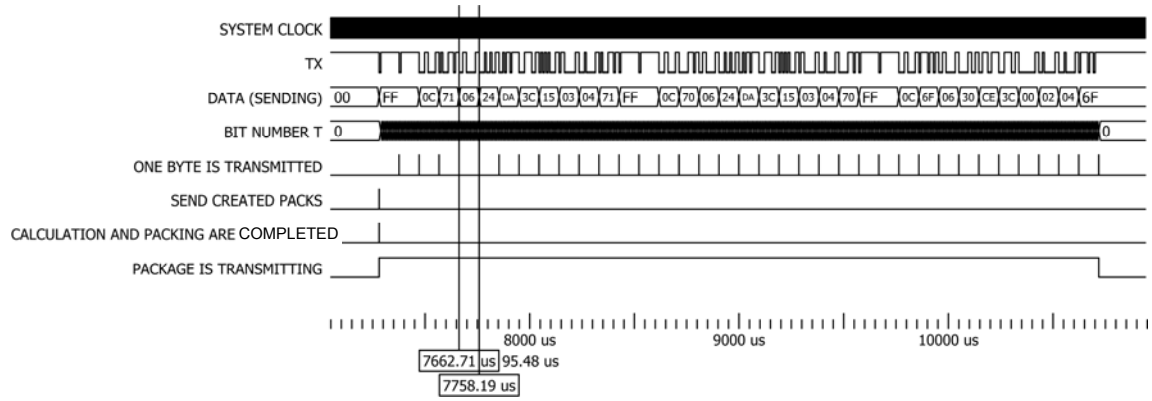

Fig. 11 Transmission of the generated packet

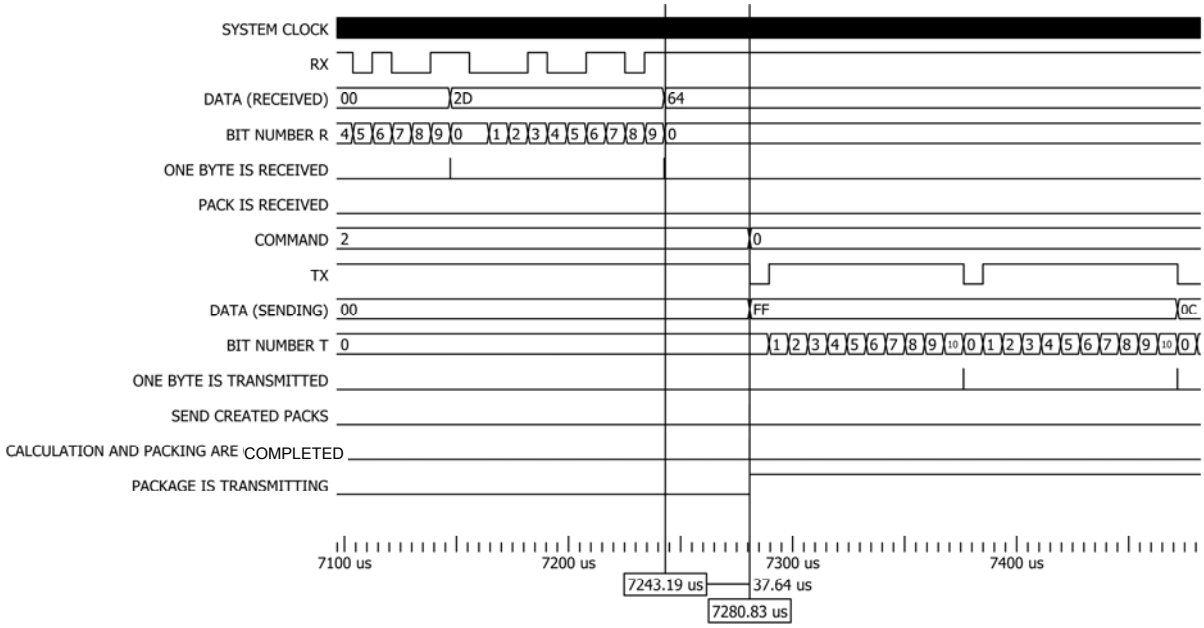

Fig. 12 Calculation period

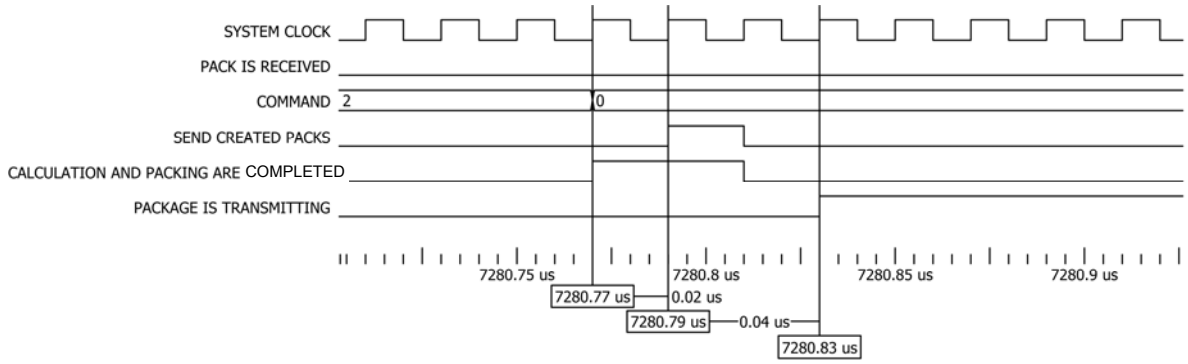

Fig. 13 'CALCULATION AND PACKAGING ARE COMPLETED', 'SEND CREATED PACKS', and 'PACKAGE IS TRANSMITTING' signals

The embedded program runs on a digital signal processor (DSP) card. Output signals of hardware were taken by a terminal program called 'Real Term' and have been displayed as a graph using the Matlab/Simulink program. The test hardware was given in Fig. 14. The DSP card sends the foot positions to the FPGA. The mainboard supplied the power requirement of all components. A universal serial bus (USB) to serial port converter was used for connection with the computer.

6.1 Input generation

Fig. 15 shows the generated input signals for the hardware test. These values were sent to the FPGA via the serial port by DSP. Fig. 16 shows the joint angles computed by Matlab/Simulink. Fig. 17 comparatively shows the outputs generated by the FPGA and simulations. The largest difference that is smaller than 1º. The test was performed using 50 commands per second.

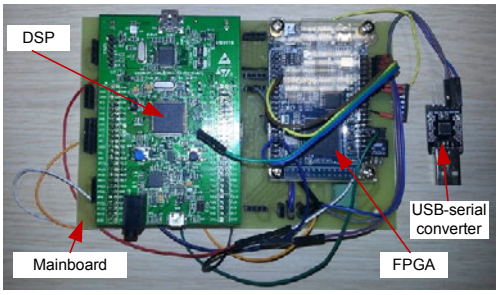

Fig. 14 Test hardware

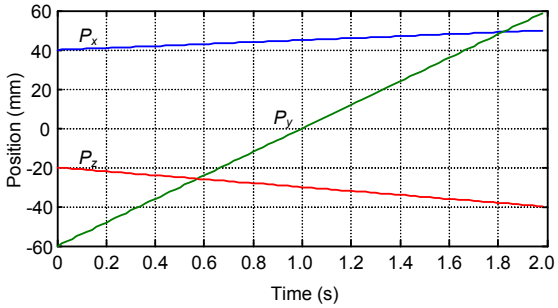

Fig. 15 x, y, and z positions of the foot

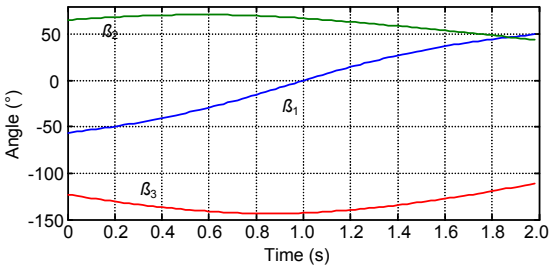

Fig. 16 Changes in joint angles

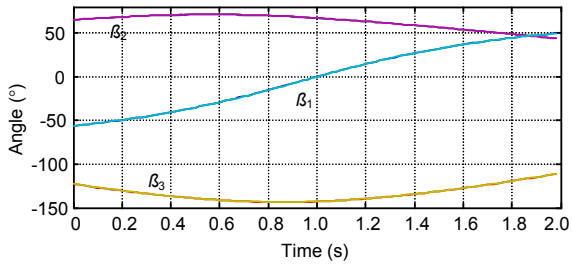

Fig. 17 Comparison of the performed design's output signals with simulation

7 Conclusions

At the end of the study, hardware has been designed and implemented. This hardware calculates the required joint angles to reach the foot positions received from the serial port. Then it sends the angles to the servo motors via the serial port. The designed hardware has a serial port and also a co-processor to perform kinematic calculations. The application was performed on FPGA to meet the needs of real-time operations. Positions of the foot are generated by an external DSP and sent to the FPGA via the serial port. Hardware has been developed for a six-legged robot. However, it can be used easily for another robot by changing the kinematic equations solved by the co-processor. Using this hardware, the transition will be faster from design to implementation in experimental or commercial studies and the load on the main processor will be reduced.

References

Axelson, J., 2007. Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems (2nd Ed.). Lakeview Research.

Barron-Zambrano, J.H., Torres-Huitzil, C., Girau, B., 2012. Configurable embedded CPG-based control for robot locomotion. *Int. J. Adv. Robot. Syst.*, **9**:92.1-92.12. [doi:10.5772/50985]

Erkol, H.O., Demirel, H., 2013. A serial port hardware design and application by FPGA. Turkish National Meeting on Automatic Control, p.132-135 (in Turkish).

Fang, Y., Chen, X., 2011. Design and simulation of UART serial communication module based on VHDL. Proc. 3rd

Int. Workshop on Intelligent Systems and Applications, p.1-4. [doi:10.1109/ISA.2011.5873448]

Fongjun, T., Tantaworrasilp, A., Kwansud, P., *et al.*, 2011. Automatic multi channel serial I/O interface using FPGA. Proc. SICE Annual Conf., p.864-867.

Hani, M.K., Wen, H.Y., Paniandi, A., 2006. Design and implementation of a private and public key crypto processor for next-generation IT security applications. *Malaysian J. Comput. Sci.*, **19**(1):29-45.

Idris, M.Y.I., Yaacob, M., Razak, Z., 2006. A VHDL implementation of UART design with BIST capability. *Malaysian J. Comput. Sci.*, **19**(1):73-86.

Juang, Y.S., Sung, T.Y., Ko, L.T., *et al.*, 2013. FPGA implementation of a CORDIC-based joint angle processor for a climbing robot. *Int. J. Adv. Robot. Syst.*, **10:**195.1-195.6. [doi:10.5772/53377]

Mahapatra, A., Roy, S.S., 2009. Computer aided dynamic simulation of six-legged robot. *Int. J. Recent Trends Eng*., **2**(2):146-151.

Murray, R.M., Li, Z., Sastry, S.S., 1994. A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton, Florida, USA.

Pa, P.S., Wu, C.M., 2012. Design of a hexapod robot with a servo control and a man-machine interface. *Robot. Comput.-Integr. Manuf.*, **28**(3):351-358. [doi:10.1016/j.rcim.2011.10.005]

Roennau, A., Kerscher, T., Dillmann, R., 2010. Design and kinematics of a biologically-inspired leg for a six-legged walking machine. Proc. 3rd IEEE RAS and EMBS Int. Conf. on Biomedical Robotics and Biomechatronics, p.626-631. [doi:10.1109/BIOROB.2010.5626328]

Sandoval-Castro, X.Y., Garcia-Murillo, M., Perez-Resendiz, L.A., *et al.*, 2013. Kinematics of hex-piderix—a six-legged robot—using screw theory. *Int. J. Adv. Robot. Syst.*, **10**:19.1-19.8. [doi:10.5772/53796]

Shih, T.S., Tsai, C.S., Her, I., 2012. Comparison of alternative gaits for multiped robots with severed legs. *Int. J. Adv. Robot. Syst.*, **9**:157.1-157.8. [doi:10.5772/52083]

Siciliano, B., Sciavicco, L., Villani, L., *et al.*, 2009. Robotics Modelling, Planning and Control. Springer, London, UK. [doi:10.1007/978-1-84628-642-1]

Taira, T., Kamata, N., Yamasaki, N., 2005. Design and implementation of reconfigurable modular humanoid robot architecture. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, p.3566-3571. [doi:10.1109/IROS.2005.1545122]

Zheng, Y., Liu, J., Kan, J., 2012. An optimal kinematics calculation method for a multi-DOF manipulator. *Przegląd Elektrotechn.*, **88**(7b):320-323.

Zhu, W., Lamarche, T., Dupuis, E., *et al.*, 2013. Precision control of modular robot manipulators: the VDC approach with embedded FPGA. *IEEE Trans. Robot.*, **29**(5):1162-1179. [doi:10.1109/TRO.2013.2265631]