Industrial Robot Arm Controller Based on Programmable System-on-Chip Device

Vo Duy Cong
Lecturer
Industrial Maintenace Training Center
Ho Chi Minh City University of Technology
268 Ly Thuong Kiet, District 10, Ho Chi Minh City, Vietnam
Vietnam National University Ho Chi Minh City
Linh Trung Ward, Thu Duc District
Ho Chi Minh City
Vietnam

Field-programmable gate arrays (FPGAs) and, recently, System on Chip (SoC) devices have been applied in a wide area of applications due to their flexibility for real-time implementations, increasing the processing capability on hardware as well as the speed of processing information in real-time. The most important applications based on FPGA/SoC devices are focused on signal/image processing, Internet of Things (IoT) technology, artificial intelligence (AI) algorithms, energy systems applications, automatic control and industrial applications. This paper develops a robot arm controller based on a programmable System-On-Chip (SoC) device that combines the highperformance and flexibility of a CPU and the processing power of an FPGA. The CPU consists of a dual-core ARM processor that handles algorithm calculations, motion planning and manages communication and data manipulation. FPGA is mainly used to generate signals to control servo and read the feedback signals from encoders. Data from the ARM processor is transferred to the programmable logic side via the AXI protocol. This combination delivers superior parallelprocessing and computing power, real-time performance and versatile connectivity. Additionally, having the complete controller on a single chip allows the hardware design to be simpler, more reliable, and less expensive.

Keywords: *industrial robot arm, robot controller, motion control system, all programmable SoC, FPGA.*

1. INTRODUCTION

Because of the ability to perform dangerous, dirty and /or repetitive tasks with consistent precision and accur–acy, the industrial robot arm is increasingly used in a variety of industries and applications such as handling, palletizing, cutting, finishing, sealing and gluing, spraying, welding… [1-6].

Controller is the core of an industrial robot arm system [7]. The robot controller plays an important role in ensuring accuracy and directly affects the speed of the robot's movement. In recent years, many studies have focused on improving robot controllers both in hardware and software. The software includes algo–rithms that perform kinematics calculations, trajectory planning, communication… and the hardware platform refers to hardware circuit [8]. A controller must have the computational ability to perform trigonometric functions in the inverse kinematics problem, trajectory interpolation, synchronous motion control. Additionally, the control system must be robust and scalable while allowing for future system improvements and expansion. Nowadays, with the development of embedded processors, complex control algorithms can be implemented by ARM processors instead of basing on industrial computers or PCI cards. Moreover, field programmable gate array (FPGA) technology is growing and being used more and more in motion controllers. In [9], the FPGA is used to control servo motors with the control signal is sent from the ARM processor. In this paper, the whole controller includes a core board and an interface board. The core board contains two main chips: STM32F207 ARM processor and Itera cyclone II EP2C8Q208 FPGA chip. The communication between these two chips is done by the FSMC (Flexible Static Memory Controller) technology which is a unique technology in STM32. A multifunctional robot arm control system based on Linux and FPGA is presented in [8]. This paper adopts IPC as the controller and FPGA as the core processor of the motion control board. Ligong-Sun in [10] designs a kind of open robot controller combining powerful functions and remarkable advantages, which realizes a programmable controller on chip of industrial robot using ALTERA FPGA and an embedded soft-core processor of NIOS II. In [11], a new motion control IC is developed andimplemented on an industry standard FPGA provided by Xilinx. This research develops functions of closed current loop control, closed position/velocity loop control, incremental encoder logic, PWM modulation, fault/brake logic, velocity estimator, host communication module, UART module and delta-sigma Analog to Digital converter. The hardware system executes quickly in dedicated parallel hardware, so the update rates of the current control loop

Received: June 2021, Accepted: September 2021
Correspondence to: Vo Duy Cong, Industrial Maintenace Training Center, Ho Chi Minh City University of Technology, Vietnam.
E-mail: congvd@hcmut.edu.vn
doi:10.5937/fme2104025C

and position/velocity control loop can reach 120 kHz and 20 kHz, respectively.

The rapid development of modern electronic technology, especially the presence of System-on-Chip devices provides a new method to implement the robot controller. The system-on-chip (SoC) devices combine high-performance CPUs and the processing power of programmable logic which delivers superior parallel-processing, computing power, real-time performance and versatile connectivity [13]. Barrios-dV et. al. in [12] design and implement a real-time controller system for robot navigation using a Xilinx Zynq® System on Chip. The system consists of a decentralized neural inverse optimal controller, an inverse kinematic model, and a path-planning algorithm. The motor control is obtained based on a discrete-time recurrent high order neural network trained with an extended Kalman filter, and an inverse optimal controller designed without solving the Hamilton Jacobi Bellman equation. In [14], a compact CNN accelerator for the IoT endpoint System-on-Chip (SoC) is proposed to meet the needs of CNN computations. The results show that the compact accelerator proposed in this paper makes the CNN computational power of the SoC based on the Cortex-M3 kernel two times higher than the quad-core Cortex-A7 SoC and 67% of the computational power of eight-core Cortex-A53 SoC.

This paper develops a robot controller using a Zynq-7000XC7Z020SoC device. The XC7Z020 devices are equipped with a dual-core ARM Cortex-A9 processor integrated with 28 nm Artix-7 based programmable logic for excellent performance-per-watt and maximum design flexibility. With the dual-core processor, the controller can implement two parallel tasks. The main core process algorithm calculations include kinematics algorithm and trajectory planning. The algorithms are executed when receiving commands from the teach pendant. Another core handles the communication and processing data from teach pendant which includes decoding robot language and G-code. After implementing the calculation, the controller needs to send signals to drivers to control servo motors via a motion control board. Because the motion control board synchronously controls six servo motors which requirea maximum frequency of up to 4Mpps, we utilize parallel computing to shorten the control cycle based on FPGA.

The remainder of this paper is structured as follows: Section 2 introduces the trajectory planning and motion control algorithms to control the robot arm. Section 3 describes hardware structure. Section 4 describes software design on ARM core and FPGA. Section 5 shows the experimental results. Finally, Section 6 is conclusion.

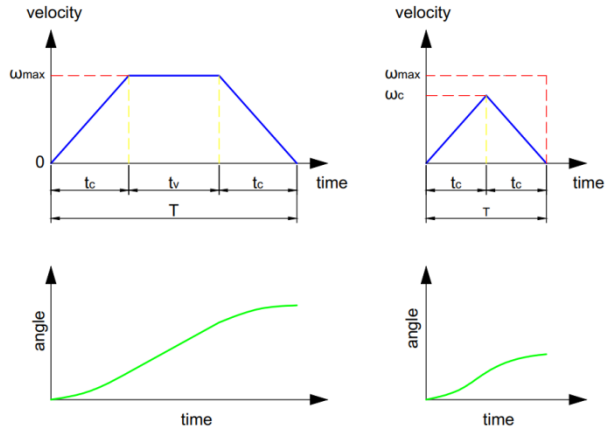
2. TRAJECTORY PLANNING AND MOTION CONTROL

Trajectory planning creates reference signals for the robot controller so that the robot moves in the desired trajectory. Motion control uses interpolation functions and inverse kinematics to generate discrete data of joint angle values. The controller sends these values to the motion board to generate pulses for servo motor drivers. The driver ensures that the motor turns properly at the reference value.

Trajectory planning generates a time schedule for how to follow a path given constraints such as position, velocity, and acceleration. The trajectory planning can be conducted in the joint space or in the workspace. The trajectory can be defined by the initial point and final point of the path (point-to-point) or a finite sequence of points along the path. In this section, we just consider the point-to-point trajectory planning problem.

2.1 Joint Space Trajectories Planning

Planning trajectory in the joint space has many advantages such as less computation, easier to plan trajectories in real-time and no problem with singularities.The initial and final pose of the robot is usually given in the workspace, solving an inverse kinematic problem, we determine the initial and the final angles at each joint.The planning algorithm generates a function *q(t)* interpolating the given joint variables at each joint. In practice, a trapezoidal velocity profile is usually assigned (see Figure 1.a). The velocity graph consists of three phases namely constant acceleration, constant velocity and constant dece–le–ration. Assume that the angle from the initial position to the final position, q_f, the maximum speed ω_{max} and the constant acceleration/deceleration $\dot{\omega}$ are given in advance. We need to determine the acceleration/ decelerationtime t_c, the time for the constant velocity phase t_v, the total time T, and the function of *q(t)*.



a. trapezoidal profile **b. triangular profile**

Figure 1. Velocity profile

The velocity at the end of the acceleration phase is equal to the constant velocity, so:

$$t_c = \frac{\omega_{\max}}{\dot{\omega}} \quad (1)$$

And the angle after the acceleration is:

$$q_c = \frac{1}{2}\dot{\omega}t_c^2 = \frac{\omega_{\max}^2}{2\dot{\omega}} \quad (2)$$

The area of trapezoid is equal to the total angle q_f, so:

$$(t_c + t_v)\omega_{\max} = q_f \quad (3)$$

Therefore, the time for the constant velocity phase is:

$$t_v = \frac{q_f}{\omega_{\max}} - \frac{\omega_{\max}}{\dot{\omega}} \tag{4}$$

If $t_v > 0$ or $q_f \cdot \dot{\omega} > \omega_{\max}^2$, the velocity profile is a trapezoid, the total time is:

$$T = 2t_c + t_v = \frac{\omega_{\max}}{\dot{\omega}} + \frac{q_f}{\omega_{\max}} \tag{5}$$

The trajectory is formed by a linear segment con–nected by twoparabolic segments:

$$q(t) = \begin{cases} \frac{1}{2}\dot{\omega}t^2 & 0 \le t \le t_c \\ \frac{\omega_{\max}^2}{2\dot{\omega}} + \omega_{\max}(t - t_c) & t_c \le t \le T - t_c \\ q_f - \frac{1}{2}\dot{\omega}(T-t)^2 & T - t_c \le t \le T \end{cases} \tag{6}$$

If $t_v < 0$ or $q_f \cdot \dot{\omega} < \omega_{\max}^2$,, the velocity profile is a triangle that only consists of acceleration and decele–ration phase (see Figure 1.b). The trajectory is formed by twoparabolic segments, we have:

$$\frac{q_f}{2} = q_c = \frac{1}{2}\dot{\omega}t_c^2 \tag{7}$$

Therefore, the acceleration/deceleration time and the total time are:

$$t_c = \sqrt{\frac{q_f}{\dot{\omega}}} \tag{8}$$

$$T = 2t_c = 2\sqrt{\frac{q_f}{\dot{\omega}}} \tag{9}$$

The maximum velocity in this case is:

$$\omega_c = \dot{\omega}t_c = \sqrt{q_f \dot{\omega}} \le \omega_{\max} \tag{10}$$

The function of angle in term of time t:

$$q(t) = \begin{cases} \frac{1}{2}\dot{\omega}t^2 & 0 \le t \le t_c \\ q_f - \frac{1}{2}\dot{\omega}(T-t)^2 & t_c \le t \le T \end{cases} \tag{11}$$

We get six values of T_i for six joints but all joints must be stopped at the same time, so the maximum value of T_i is selected:

$$T = \max_{i=1,6}(T_i) \tag{12}$$

Then, the velocity $\omega_{\max}$ at each joint is recalculated by solving equation (5):

$$\omega_{\max} = \frac{T\dot{\omega} - \sqrt{(T\dot{\omega})^2 - 4q_f\dot{\omega}}}{2} \tag{13}$$

If $(T\dot{\omega})^2 - 4q_f < 0$ or $T < 2\sqrt{q_f / \dot{\omega}}$, the accelera–tion is recalculatedfrom equation (9) and the triangular profile is used:

$$\dot{\omega} = \frac{4q_f}{T^2} \tag{14}$$

2.2 Task Space Trajectories Planning

Joint space trajectories are usually used in pick and place applications that do not care about the end-effect or trajectory. But in many applications, such as welding, cutting, etc., it is requiredto control the robot motion to follow a specified path in the workspace. So, trajectory planning also needs to be executed directly in the workspace. In the task space trajectories, the position and orientation of the robot's end-effector are inter–polated over time and transformed to the values of the joint angles bysolving the inverse kinematics. So, it requires a large amount of computation.

The trajectories are usually formed by linear and circular paths. Consider the linear trajectory between the initial position p_i and final position p_f. Perform thelinear interpolation:

$$p(t) = p_i s(t)(p_f - p_i) \tag{15}$$

where s is a scalar, $s \in [0,1]$. Usually, the trajectory starts and ends withzero speed, so the trapezoidal profile is used to defined *s(t)*:

$$s(t) = \begin{cases} \frac{\dot{s}_{\max}}{2t_c}t^2 & 0 \le t \le t_c \\ \dot{s}_{\max}\left(t - \frac{t_c}{2}\right) & t_c < t \le T - t_c \\ 1 - \frac{\dot{s}_{\max}}{2t_c}(T-t)^2 & T - t_c < t \le T \end{cases} \tag{16}$$

where $\dot{s}_{\max} = 1/(T - t_c)$ is the maximum value of $\dot{s}(t)$. The values of t_c and T are calculated from acceleration a_m and velocity v_m that are given in advance (similar as equation 1 and equation 5):

$$\begin{aligned} v_m &= a_m t_c \\ T &= \frac{a_m}{v_m} + \frac{l}{v_m} \end{aligned} \tag{17}$$

where l is the length of the line segment.

In the case of the circular trajectory with radius r, center **C** and orientation matrix **R**, the coordinate of a point **P**on the circle is:

$$P(t) = c + R\begin{bmatrix} r\cos(s(t)/r) \\ r\sin(s(t)/r) \\ 0 \end{bmatrix} \tag{18}$$

where *s(t)* is the arc length. The value of *n(t) = s(t)/r*is also computed as in the linear interpolation.

When robot movement, the orientation of the end-effector is also changed.The orientation of the end-effector is represented by the rotation matrix. Denote the rotation matrix at the initial position is $\boldsymbol{R}(t=0) = \boldsymbol{R}_i$ and at the final position is $\boldsymbol{R}(t=T) = \boldsymbol{R}_f$. To perform the orientation planning, define the matrix $\boldsymbol{R}^f_i$ that trans–forms the initial orientation to the final orientation. We have the relationship $\boldsymbol{R}^f = \boldsymbol{R}_i\boldsymbol{R}^f_i$ or:

$$R_i^f = R_i^T R_f \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{19}$$

Calculate the angle-axis parameter $\boldsymbol{u}\theta$ from the rota–tion matrix $\boldsymbol{R}^f_i$:

$$\theta = \cos{-1}\left(\frac{r_{11}+r_{12}+r_{13}-1}{2}\right)$$
$$u = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix} \tag{20}$$

Choose the angle $\alpha(t)$ so that $\alpha(t=0) = 0$, $\alpha(t=T) =$ (and use a trapezoidal profile) to calculate the matrix:

$$R_t = \begin{bmatrix} r_x^2 a + c\alpha & r_x r_y a - r_z s\alpha & r_x r_z a - r_y s\alpha \\ r_x r_y a + r_z s\alpha & r_y^2 a + c\alpha & r_y r_z a - r_x s\alpha \\ r_x r_z a - r_y s\alpha & r_y r_z a - r_x s\alpha & r_z^2 a + c\alpha \end{bmatrix} \tag{21}$$

with $c\alpha = \cos\alpha$, $s\alpha = \sin\alpha$, $a = 1 - c\alpha$ (21)

Then, calculate the orientation matrix of the end-effector:

$$R(t) = R_i R_t \tag{22}$$

2.3 Pulse train generation

Combining the trajectory planning and inverse kine–matics, the joint angles at each joint can be generated over time. The motion control system needs to control the servo motors to reach these values. A servo motor is controlled by sending pulses to the motor driver. One pulse makes the motor rotate one small constant angle called resolution. The pulses are generated every sampling cycle T. Assuming $q(t)$ is the joint angle at the time t and $q(t+T)$ is the joint angle at the next period. The number of pulses sent to the controller is determined:

$$n = \begin{cases} 0 & ,if\left|q(t+T)-q(t)\right| < resolution \\ \left[\dfrac{q(t+T)-q(t)}{resolution}\right] & ,if\left|q(t+T)-q(t)\right| \geq resolution \end{cases} \tag{23}$$

The number of pulses is sent to the motion controller and the motion controller ensures to generate pulses at the same time, so six servo motors will work synch–ronously. When pulses are generated independently on the motion controller, the pulse frequency can be up to several MHz so the motor can turn at high resolution and leads to many benefits such as: higher efficiency, less heat, increased stability, better speed control, higher torque to inertia ratio possibilities.

3. HARDWARE DESIGN

The integration of two ARM cores and FPGA on a single device helps to simplify the hardware structure. This research uses the MicroZed board as a central processor board. MicroZed contains two I/O headers that provide the connection to two I/O banks on the programmable logic (PL) side of the Zynq®-7000 All Programmable SoC device. An external board is desi–gned to connect MicroZed board with servo driver thro–ugh these I/O headers. In addition, the external board is also responsible for converting 2.5V signalson the MicroZed board to 5V signalson the servo driver and vice versa. The hardware design is shown in Figure 2.

Unlike traditional robot controllers, the main cont–roller and motion controller are distinguished, our controller is designed on a single board. So, the size of the controller is very compact and the communication is also simple. Although integrated into a single board, the functions of each part are still the same as the previous controllers. The main core ARM is mainly responsible for algorithm calculations, trajectory planning, commu–nication with teach pendant and other devices, control–ling FPGA. FPGA is used to control the servo motor and read the feedback signals from the encoder. More–over, FPGA provides reprogrammable IO and hardware reconfiguration capability, so we can expand IO pins to control the end-effector tools and create more hardware for communication standards. The main controller will manage this IO pin function and create software to control communications.

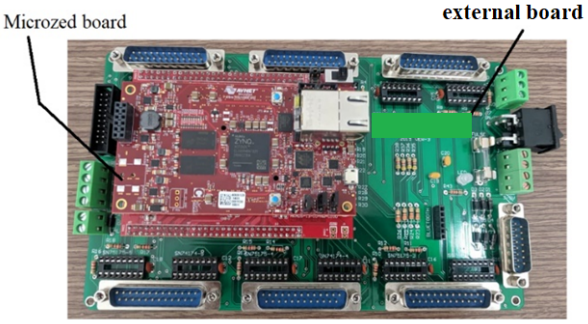


Figure 2. Hardware design

Another advantage of the Zynq All Programmable SoC device is communication between ARM core and FPGA. Data is communicated via the AXI bus inter–connect. Advanced Extensible Interface, or AXI, is part of ARM's AMBA specifications. The AXI is a point-to-point interconnect that is designed for high-perfor–mance, high-speed microcontroller systems. The AXI protocol is based on a point-to-point interconnect to avoid bus sharing and therefore allow higher bandwidth and lower latency. The Xilinx AXI Interconnect contains AXI-compliant master and slave interfacesand can be used to route transactions between one or more AXI masters and slaves. In our system, the ARM core is the master, slaves are modules designed on the FPGA side, which means one master interfaces with many slaves. Each slave device must have a unique address.

When the ARM chip writes data to FPGA, it sends address and control signals to slaves. The control signal can be ‘read’ or ‘write’. After that, the master sendsthe channel address to choose which channel will be read or write. Each channel is a 32-bit register. If the control signal is ‘read’, the slave will send data to the master, otherwise if the control signal is ‘write’, the master will send data to the slave.

4. SOFTWARE DESIGN

4.1 FPGA Design

To accelerate the creation of highly integrated and complex designs in programmable devices, we use intelligent IP integration delivered by the Vivado Design Suite software. An IP (intellectual property) core is a block of logic or data that is used in making anFPGA or application-specific integrated circuit (ASIC) for a product. Figure 3 shows the diagram designed using IP blocks, in which some blocks are pre-defined in the software, the rest are user-defined blocks. The user-defined blocks are programmed by VHDL language. The functions of the IP blocks are as follows:

- ZYNQ7 Processing System IP: the software interface around the Zynq-7000 Processing System.
- AXI Interconnect IP: connects one or more AXI memory-mapped masterdevices to one or more memory-mapped slave devices
- AXI Timer IP: create a trigger signal to synch–ronize data processing and communication between ARM processor core and FPGA. The trigger signal is created after every 50us that is configured by the software.
- AXI UART16550 IP: provide the controller interface for asynchronous serial data transfer. An interrupt signal at the ip2intc_irpt pin is created after receiving enough 8 bytes of data.
- Pulse_train_v1.1 IP (user define IP): includes six pulse generator modules that generate the pulse signals for six servo motors. All the pulse generator modules are triggered by the trigger signal from the AXI timer so that they can start to generate pulses at the same time.
- switch_v1.0 and encoder_v1.1 IP (user define IP): reads signal from encoder. Encoder IP only reads one encoder once, so switch IP is designed to choose the encoder.

The encoder IP consists of Debounce Circuit and Decoder Circuit as shown in Figure 4 and Figure 5. The debounce circuit is used to reduce noise when the signal level changes.FF1 and FF2 always store the last two logic levels of the encoder signal.If the signal’slevel changes, the values of FF1and FF2 are differentin a clock cycle, the output of the XOR logic gate is HIGH and reset the counter. If the signal’s level is unchanging, the output of the XOR logic gate is LOW and the counter begins to count. If the signal’s logic level is stable, the counter continues to increment until it reaches the debounce_time value. The FF3 is enabled and the stable value is clocked through to the output. The counter disables itself until there is another change on one of the inputs.

Based on the operation of quadrature encoders [16], Table 1 lists the possible a and b input transitions, along with the values of direction and position. As seen in the truth table, the direction is determined by:

Direction = a_new XOR b_prev

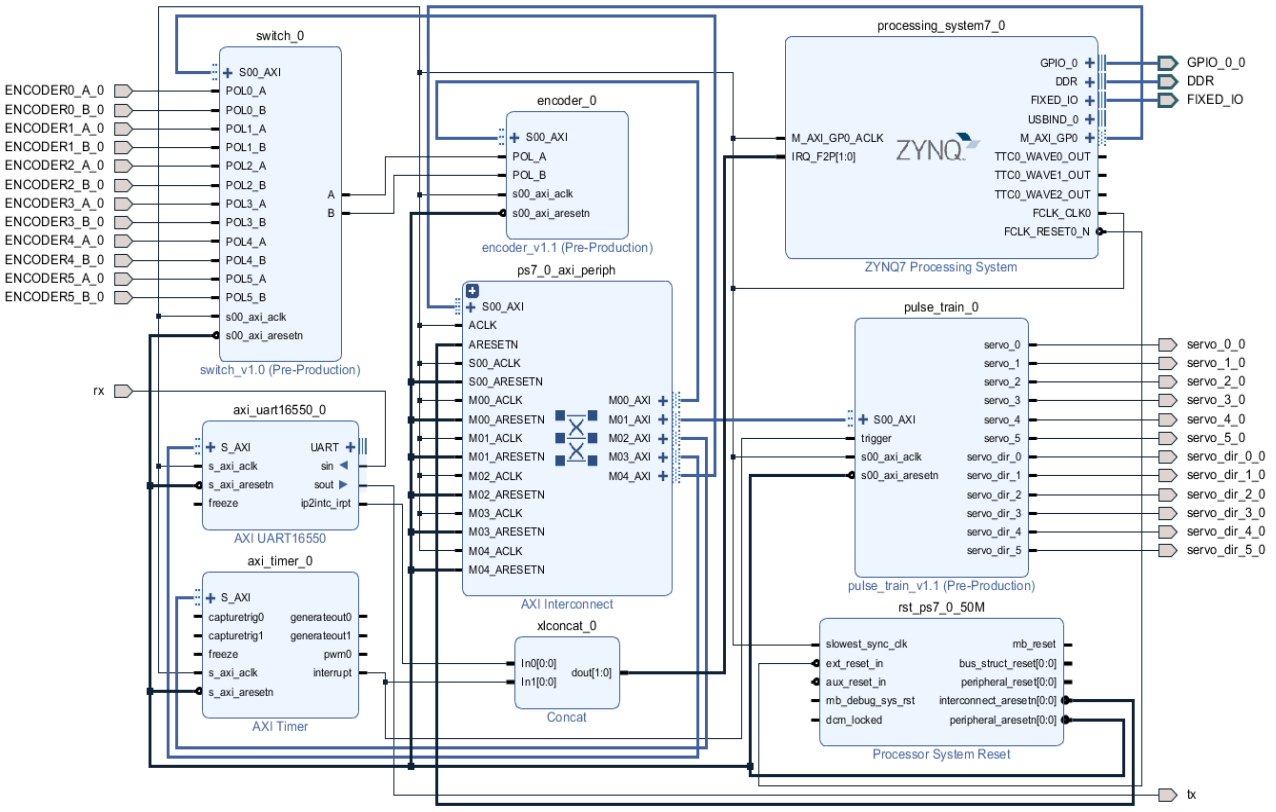


Figure 3. Block IP design

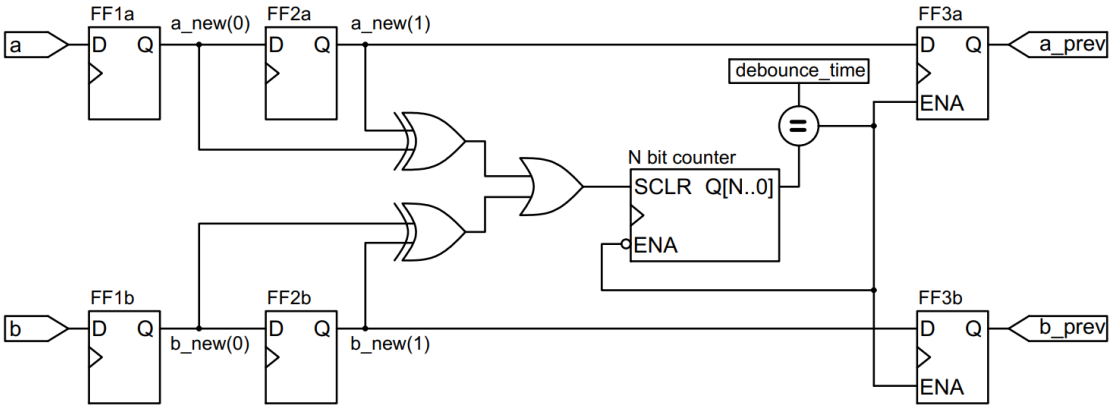


Figure 4. Synchronization and Debounce Circuit

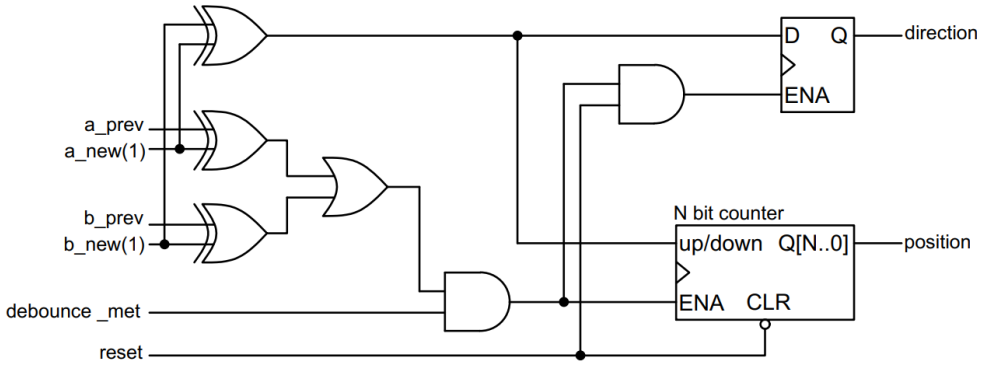


Figure 5. Decoder Circuit

Table 1: Truth Table

Previous Inputs		New Inputs		Results	
a_prev	b_prev	a_new	b_new	Direction	Position
0	0	1	0	1	increase
1	0	1	1	1	increase
1	1	0	1	1	increase
0	1	0	0	1	increase
0	0	0	1	0	decrease
0	1	1	1	0	decrease
1	1	1	0	0	decrease
1	0	0	0	0	decrease

The value of the counter only changes (increase or decrease) when the debounce time is met (stable signal) and has a rising edge or a falling edge of the encoder pulse. The signal edge is detected by using an XOR logic gate.

The algorithm to generate the pulse signal for servo motors in the Pulse_train IP is illustrated in Figure 6. The value of pulse width is sent from the ARM processor through AXI interconnect. The direction of rotation depends on the sign of the pulse width and is outputted to servo_dir pins. The pulse signal is generated on servo_i pins. S_AXI_ACLK is a synchronous signal for AXI communication and it is used to change the value of the "count" variable. The frequency of S_AXI_ACLK is set to 50 MHz.

4.2 ARM Software Design

The software will be divided into several smaller tasks. The tasks are performed depending on the command received from a teach pendant. After the ARM processor core starts, it conducts hardware initialization such as UART, Timer, GPIO and so on. Then, it will wait for the teach pendant commands to perform the tasks. Tasks can be online or offline.

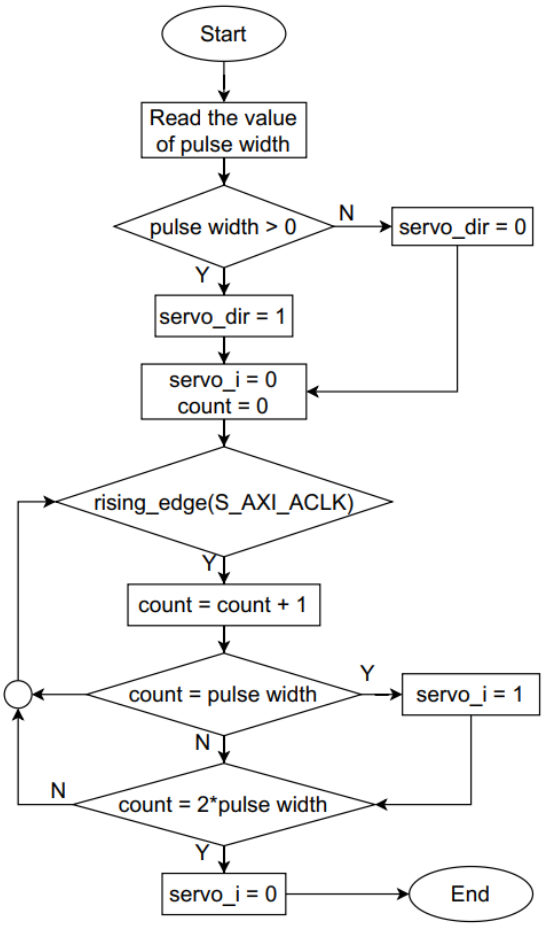


Figure 6. Pulse generation algorithm

The online task means that after receiving a teach pendant command, the controller will immediately execute this command. Then, it waits for the next com–mand. The online tasks include: set coordinate system, change speed, jogging commands, interpolation com–mands, …

In the offline mode, the robot will receive a program that is written in the robot language, save the program in the memory. Then, the controller will encode and execute each command in the program. While implementing the program, except for an emergency stop command, all commands from the teach pendant will not be executed. Because we use the UART interrupt, the controller always receives data from the teach pendant, but when a program is running, only the emergency stop command is executed.

5. EXPERIMENT RESULTS

Our controller is used to control a Six-DOF robot arm which is shown in Figure 7. The controller receives commands from a teach pendant that is designed on a Samsung tablet (Figure 8). The robot canbe controlled to perform tasks such as pick and place, laser cutting, point welding, linear welding…

Figure 7. Six DOF robot arm

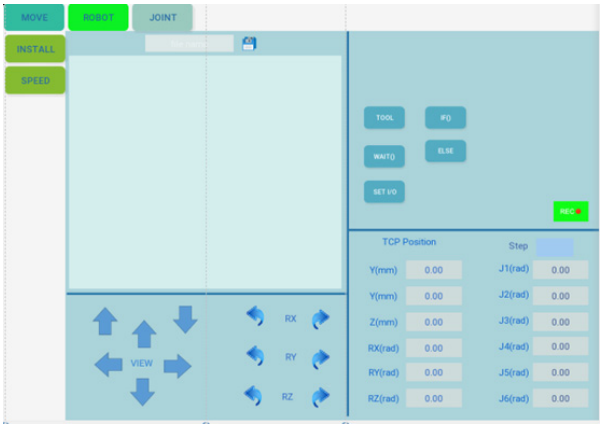


Figure 8. Teach pendant interface

Figure 9 shows the results when applying the trapezoidal velocity profiles to create the angled command at each joint. All six joints are started and finished at the same time with a total time of 0.87s. The command angles of joint 4 and joint 5 aremuch smaller than the other joints, so they move with the triangle profiles with the maximum velocity of 40 deg/s.It should be noted that the motor is connected to the gearbox 50:1 ratio. The acceleration of joint 6 is set slightly larger than other joints.

Figure 9shows the results of using the robot to draw some basic shapes.It can be seen that the robot can imp–lement interpolation in the workspace and move smoothly.

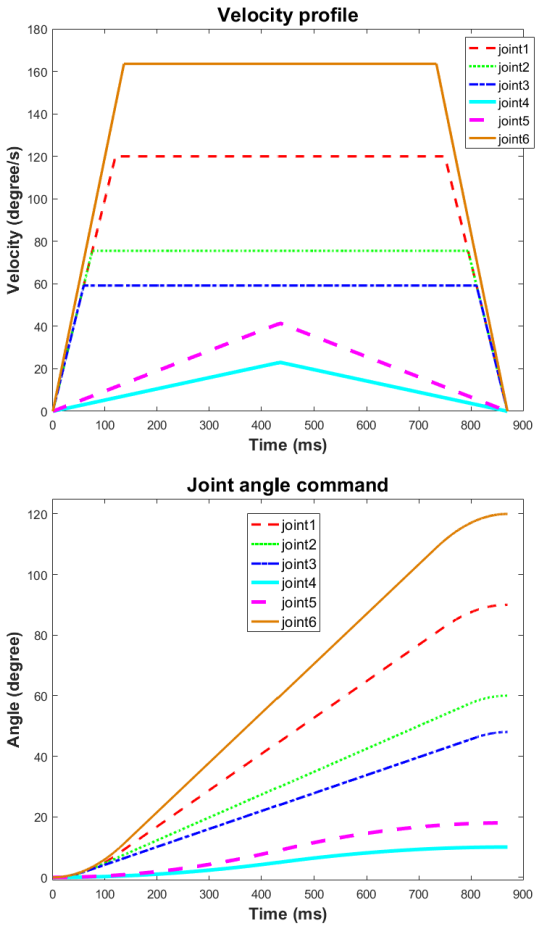


Figure 9. Joint angle trajectory planning

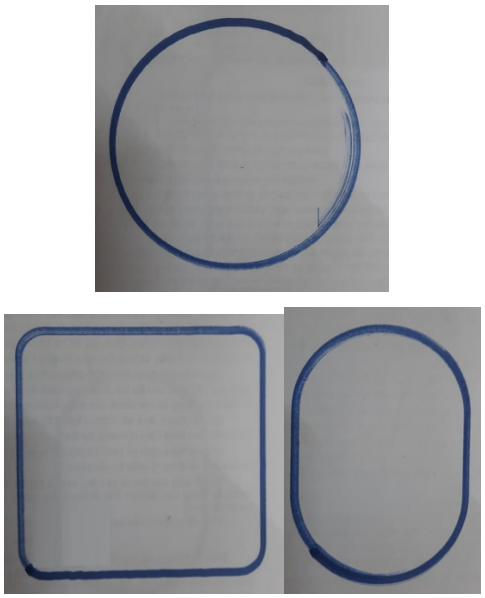


Figure 10. Robot movement results

6. CONCLUSION

In this paper, the 6 DOF industrial robot arm controller is designed based on the Zynq All Programmable SoC

device. The main controller and motion controller are designed on a single board. So, the size of the controller is very compact, simple, low-cost and low power con–sumption.

The developed controller incorporates ARM's com–putational power and parallel processing capabilities of the FPGA, easily communicating with teach pendant and other peripherals. The ARM microprocessor was incorporated to take advantage of the high-level programming for algorithm calculations include kinematics algorithm and trajectory planning, handling the communication and processing data from teach pendant which inc–ludes decoding robot language and G-code. The FPGA-based motor motion control has given synchronously control of six servo motors which require a maximum frequency of up to 4Mpps so the motor can turn at high resolution and resulting in higher efficiency, less heat, increased stability, better speed control, higher torque to inertia ratio possibilities. Moreover, FPGA provides reprogrammable IO and hardware reconfiguration capability, so we can expand IO pins to control the end-effector tools and create more hardware for communication standards.

The experimental results show that the controller can control the robot to implement interpolation in both joint space and workspace leading to smooth move–ments. The robot can be used to perform most of the common tasks in industrial production.

In future work, AI algorithms will be integrated into our robot controller. AI algorithms are often used for robot motion planning in high-level control and artificial neural networks can be implemented with an FPGA in a System on Chip (SoC) device. The goal of an artificial neural network in the context of robot arm control is to train a deep policy neural network to create the optimal sequence of motion commands. The output of this policy network is torques or velocity commands for each actuator.

APPENDIX

ROBOT KINEMATICS

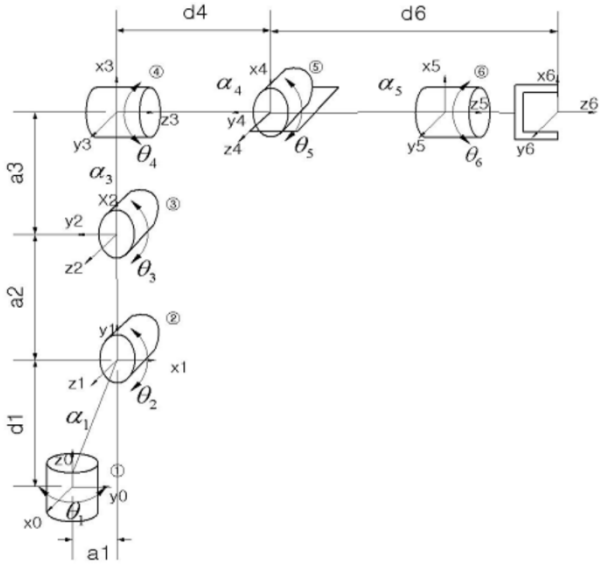


Figure 11. Robot coordinates

Transformation between two joints in a generic form is given by [15]:

$$T_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i s\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Table 2: DH parameters

Link	a_i (mm)	α_i (deg)	d_i (mm)	θ_i (deg)
1	$\mathbf{a_1}$	**90**	$\mathbf{d_1}$	θ_1
2	$\mathbf{a_2}$	**0**	**0**	θ_2
3	$\mathbf{a_3}$	**90**	**0**	θ_3
4	**0**	**-90**	$\mathbf{d_4}$	θ_4
5	**0**	**90**	**0**	θ_5
6	**0**	**0**	$\mathbf{d_6}$	θ_6

Robot coordinates shown in Figure 11are used to establish DH parameters (Table 2). The DH parameters are substituted into equation (24) to find the transfor–mation matrices, from ${}_0^1T$ to ${}_5^6T$. The total trans–formation matrix between the base of the robot and link 6 is:

$$\begin{aligned} V_0 &= U_1 = {}_6^0T = {}_1^0T_2^1T_3^2T_4^3T_5^4T_6^5T \\ V_1 &= U_2 = \left({}_1^0T\right)^{-1} \cdot {}_6^0T = T_3^2T_4^3T_5^4T_6^5T \\ V_2 &= U_3 = \left({}_2^1T\right)^{-1} \cdot V_1 = T_4^3T_5^4T_6^5T \\ V_3 &= U_4 = \left({}_3^2T\right)^{-1} \cdot V_2 = T_5^4T_6^5T \\ V_4 &= U_5 = \left({}_4^3T\right)^{-1} \cdot V_3 = T_5^4T_6^5 \\ V_5 &= U_6 = \left({}_5^4T\right)^{-1} \cdot V_4 = {}_6^5T \end{aligned} \tag{26}$$

where r_{ii} is the orientation elements and (q_x,q_y,q_z) is the position elements. These elements are shown in the following equations:

$$\begin{aligned} r_{11} &= s_6\left(s_1c_4 - c_1s_4c_{23}\right) - c_6\left[c_1s_5s_{23} - c_5\left(s_1s_4 + c_1c_4s_{23}\right)\right] \\ r_{12} &= c_6\left(s_1c_4 - s_1s_4c_{23}\right) + s_6\left[c_1s_5s_{23} - c_5\left(s_1s_4 + c_1c_4s_{23}\right)\right] \\ r_{13} &= c_1c_5s_{23} + s_5\left(s_1s_4 + c_1c_4c_{23}\right) \\ r_{21} &= -c_6\left[s_1s_5s_{23} + c_5\left(c_1s_4 + s_1s_4c_{23}\right)\right] - s_6\left(c_1c_4 + s_1s_4c_{23}\right) \\ r_{22} &= s_6\left[s_1s_5s_{23} + c_5\left(c_1s_4 + s_1s_4c_{23}\right)\right] - c_6\left(c_1c_4 + s_1s_4c_{23}\right) \\ r_{23} &= s_1c_5s_{23} - s_5\left(c_1s_4 - s_1s_4c_{23}\right) \\ r_{31} &= c_6\left(c_{23}s_5 + s_{23}c_4c_5\right) - s_{23}s_4s_6 \\ r_{32} &= -s_6\left(c_{23}s_5 + s_{23}c_4c_5\right) - s_{23}s_4s_6 \\ r_{33} &= s_{23}c_4s_5 - c_{23}c_5 \\ q_x &= p_x + d_6r_{13} \\ q_y &= p_y + d_6r_{23} \\ q_z &= p_z + d_6r_{33} \\ p_x &= d_4c_1s_{23} + a_3c_1c_{23} + a_2c_1c_2 + a_1c_1 \\ p_y &= d_4s_1s_{23} + a_3s_1c_{23} + a_2s_1c_2 + a_1s_1 \\ p_z &= -d_4c_{23} + a_3s_{23} + a_2s_2 + d_1 \end{aligned} \tag{27}$$

with $s_i = \sin\theta$, $c_i = \cos\theta_i$

$c_{ij} = \cos(\theta_i+\theta_j)$, $s_{ij} = \sin(\theta_i+\theta_j)$

The above equations are called forward kinematic equations that describe the relationship between theindividual joints of the robot manipulatorand the position and orientation of the end-effector. Contrarily, the inverse kinematic problemtransformsthepositionand orientationof theend-effector in the Cartesian space to the joint space.

To find the inverse kinematics solution, denote:

$$\begin{aligned}
V_0 &= U_1 = {}^0_6T = {}^0_1T\,T^1_2T^2_3T^3_4T^4_5T^5_6T\\
V_1 &= U_2 = \left({}^0_1T\right)^{-1}\cdot {}^0_6T = T^2_3T^3_4T^4_5T^5_6T\\
V_2 &= U_3 = \left({}^1_2T\right)^{-1}\cdot V_1 = T^3_4T^4_5T^5_6T\\
V_3 &= U_4 = \left({}^2_3T\right)^{-1}\cdot V_2 = T^4_5T^5_6T\\
V_4 &= U_5 = \left({}^3_4T\right)^{-1}\cdot V_3 = T^4_5T^5_6\\
V_5 &= U_6 = \left({}^4_5T\right)^{-1}\cdot V_4 = {}^5_6T
\end{aligned} \tag{28}$$

The multiplications are carried out and yield the results as follows:

$$V_1 = \begin{bmatrix} c_1V_{01} + s_1V_{02} - a_1M \\ V_{03} - d_1M \\ s_1V_{01} - c_1V_{02} \\ M \end{bmatrix} \tag{29}$$

$$V_2 = \begin{bmatrix} c_2V_{11} + s_2V_{12} - a_2M \\ s_2V_{11} + c_2V_{12} \\ V_{13} \\ M \end{bmatrix} \tag{30}$$

$$V_3 = \begin{bmatrix} c_{23}V_{11} + s_2V_{12} - (a_2c_3 + a_3)M \\ V_{13} \\ s_{23}V_{11} - c_{23}V_{12} - a_2s_3M \\ M \end{bmatrix} \tag{31}$$

$$V_4 = \begin{bmatrix} c_4V_{31} + s_4V_{32} \\ -V_{33} + d_4M \\ s_4V_{31} + c_4V_{32} \\ M \end{bmatrix} \tag{32}$$

$$V_5 = \begin{bmatrix} c_{55}V_{41} + s_5V_{42} \\ -V_{43} \\ s_5V_{41} - c_5V_{42} \\ M \end{bmatrix} \tag{33}$$

$$U_5 = \begin{bmatrix} c_5c_6 & -c_5c_6 & s_5 & 0 \\ s_5c_6 & -s_5c_6 & -c_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

$$U_4 = \begin{bmatrix} c_4U_{511} - s_4s_6 & c_4U_{512} - s_4c_6 & c_4s_5 & 0 \\ c_4U_{511} + c_4s_6 & c_4U_{512} + c_4c_6 & s_4c_5 & 0 \\ -U_{521} & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

$$U_{3j4} = \begin{bmatrix} d_4s_3 + a_3c_3 \\ -d_4c_3 + a_3s_3 \\ 0 \\ 1 \end{bmatrix} \tag{36}$$

$$U_{3j4} = \begin{bmatrix} d_4s_3 + a_3c_3 \\ -d_4c_3 + a_3s_3 \\ 0 \\ 1 \end{bmatrix} \tag{37}$$

$$U_{1j4} = \begin{bmatrix} c_1U_{214} + a_1c_1 \\ s_1U_{214} + a_1s_1 \\ 0 \\ 1 \end{bmatrix} \tag{38}$$

where V_{ij} is the jth row of matrix V_i, U_{ijk} is the element at the jth row and kth column of matrix U_i, M = [0 0 0 1]. The joint angles θ_i are solved as follows:

From two equations $V_{014} = U_{214}$ and $V_{024} = U_{124}$, solve θ_1:

$$\theta_1 = a\tan 2\left(p_y, p_x\right) \tag{39}$$

Substituting θ_1 into equation (29) to compute the matrix V_1. From two equations $V_{114} = U_{214}$ and $V_{114} = U_{214}$, we can solve θ_1:

$$\theta_6 = a\tan 2\left(V_{431}, c_5V_{411} + s_5V_{421}\right) \tag{40}$$

where:

$$n_c = \frac{a_2^2 - d_4^2 - a_3^2 + V_{114}^2 + V_{124}^2}{2a_2\sqrt{V_{114}^2 + V_{124}^2}} \tag{41}$$

Substituting θ_2 into equation (30) to compute the matrix V_2.From two equations $V_{214} = U_{314}$ and $V_{114} = U_{214}$, solve θ_2:

$$\theta_3 = a\tan 2\left(V_{214}, -V_{224}\right) - \mathrm{a}\tan 2\left(a_3, d_4\right) \tag{42}$$

From two equations $V_{313} = U_{413}$ and $V_{323} = U_{423}$, we have:

$$V_{313} = c_4s_5,\ V_{323} = s_4s_5 \tag{43}$$

If $\sin\theta_5 \neq 0 \rightarrow \theta_4 = \tan^{-1}\left(V_{323}/V_{313}\right)$

From two equations $V_{413} = U_{513}$ and $V_{423} = U_{523}$, solve θ_5:

$$\theta_5 = a\tan 2\left(c_4V_{313} + s_4V_{323}, V_{333}\right) \tag{44}$$

Finally, from two equations $V_{511} = U_{611}$ and $V_{521} = U_{621}$, solve θ_6:

$$\theta_6 = a\tan 2\left(V_{431}, c_5V_{411} + s_5V_{421}\right) \tag{45}$$

ACKNOWLEDGMENT

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

REFERENCES

[1] H. F. Fauadi and M. S. Jumali: Modelling and simulation of programmable universal machine for assembly (PUMA) industrial robot for automotive related assembly process. International Symposium on Information Technology, pp. 1-5, 2008

[2] G. S. Huang, C. K. Tung, H. C. Lin, and S. H. Hsiao.: Inverse kinematics analysis trajectory planning for a robot arm. Control Conference, pp. 965–970, 2011.

[3] E. S. Kheng, A. H. A. Hassan, A. Ranjbaran, and T. S. Siong.: Range estimation for robot arm applications using image segmentation and curve fitting tool. International Conference on Electrical, Control and Computer Engineering, pp. 275–278, 2011.

[4] X. Chu, H. Fleischer, N. Stoll, M. Klos, and K. Thurow.: Application of dual-arm robot in biomedical analysis: Sample preparation and transport. Instrumentation and Measurement Technology Conference, pp. 500–504, 2015.

[5] M. Polishchuk and M. Tkach.: Experimental Studies of Robotic Assembly of Precision Parts. FME Transactions, 2021, Vol. 49, No.1, pp. 44-55.

[6] D. Antonelli and G. Bruno.: Dynamic Distribution of Assembly Tasks in a Collaborative Workcell of Humans and Robots. FME Transactions, 2019, Vol. 47, pp. 723-730.

[7] E. Coste-Maniere and R. Simmons.: Architecture, the backbone of robotic systems. IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA, 2000, pp. 67-72,Vol.1.

[8] Cong Han, Hongbin Ma, Wenchao Zuo, Sunjie Chen, and Xinghong Zhang.: A General 6-DOF Industrial Robot Arm Control System Based on Linux and FPGA. Chinese Control And Decision Conference (CCDC), pp.1220 – 1225, 2018.

[9] Xingqiang He, Zhengdong Wang, Haitao Fang, Kai He and RuxuDu.: An Embedded Robot Controller Based on ARM and FPGA. IEEE International Conference on Information Science and Technology, pp.702-705, 2014.

[10] Ligong Suna, Fei Xiangb, Xiangwen Sunc, Sujuan Lid.: Design of Industrial Robot Controller Based on System on Programmable Chip. International Conference on Electronic & Mechanical Engineering and Information Technology, pp.3877 -3880, 2011.

[11] Shao Xiaoyin, Sun Dong.: A FPGA-based motion control IC design, IEEE International Conference on Industrial Technology, pp. 131 - 136, 2005.

[12] Barrios-dV S, Lopez-Franco M, Rios JD, Arana-Daniel N, Lopez-Franco C, Alanis AY. An Autonomous Path Controller in a System on Chip for Shrimp Robot. Electronics. 2020; 9(3):441

[13] I. Bravo-Muñoz, J.L. Lázaro-Galilea and A.Gardel-Vicente.: FPGA and SoC Devices Applied to New Trends in Image/Video and Signal Processing Fields. Electronics, 2017, Vol.6, No.25.

[14] Ge F, Wu N, Xiao H, Zhang Y, Zhou F.: Compact Convolutional Neural Network Accelerator for IoT Endpoint SoC. Electronics. 2019; 8(5).

[15] P. Corke. Robotics, Vision and Control, Fundamental Algorithms in Matlab®. Springer, 2011.

[16] Quadrature Decoder (VHDL), Digi-Key Tech Forum. https://forum.digikey.com/t/quadrature-decoder-vhdl/12671

КОНТРОЛЕР РУКЕ ИНДУСТРИЈСКОГ РОБОТА БАЗИРАН НА ПРОГРАМАБИЛНОМ SOC УРЕЂАЈУ

В.Д. Конг

FPGA и SOC уређаји имају широку употребу због своје флексибилности за примену у реалном времену, повећане снаге процесора као и брзине обраде информација у реалном времену. Најзначајнија примена базирана на FPGA/SOC уређајима се односи на обраду сигнала/слике, IoT технологије, AI, апликације енергетског система, аутоматско управљање и индустријску примену. Развијен је контролер роботске руке базиран на програмабилном SOC уређају који представља спој веће перформансе и флексибилности процесора и процесорске снаге FPGA. Процесор се састоји од двојезгреног ARM процесора који врши алгоритамска срачунавања, планирање кретања, управља комуницирањем и манипулише подацима. FPGA се претежно користи за генерисање сигнала за управљање серво системом и читање повратног сигнала са енкодера. Подаци са ARM процесора се преносе на програмабилну логичку страну преко AXI протокола. Овом комбинацијм се обавља боља паралелна обрада и брже срачунавање, она има боље перформансе и разноврсност конективности. Када је цео контролер на једном чипу могуће је дизајнирати једноставнији, поузданији и јефтинији хардвер.