Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 906505, 10 pages
http://dx.doi.org/10.1155/2015/906505

Research Article

Digital Hardware Realization of Forward and Inverse Kinematics for a Five-Axis Articulated Robot Arm

Bui Thi Hai Linh and Ying-Shieh Kung

Department of Electrical Engineering, Southern Taiwan University of Science and Technology, 1 Nan-Tai Street, Yong-Kang District, Tainan City 710, Taiwan

Correspondence should be addressed to Ying-Shieh Kung; kung@mail.stust.edu.tw

Received 16 August 2014; Accepted 13 September 2014

Academic Editor: Stephen D. Prior

When robot arm performs a motion control, it needs to calculate a complicated algorithm of forward and inverse kinematics which consumes much CPU time and certainty slows down the motion speed of robot arm. Therefore, to solve this issue, the development of a hardware realization of forward and inverse kinematics for an articulated robot arm is investigated. In this paper, the formulation of the forward and inverse kinematics for a five-axis articulated robot arm is derived firstly. Then, the computations algorithm and its hardware implementation are described. Further, very high speed integrated circuits hardware description language (VHDL) is applied to describe the overall hardware behavior of forward and inverse kinematics. Additionally, finite state machine (FSM) is applied for reducing the hardware resource usage. Finally, for verifying the correctness of forward and inverse kinematics for the five-axis articulated robot arm, a cosimulation work is constructed by ModelSim and Simulink. The hardware of the forward and inverse kinematics is run by ModelSim and a test bench which generates stimulus to ModelSim and displays the output response is taken in Simulink. Under this design, the forward and inverse kinematics algorithms can be completed within one microsecond.

1. Introduction

The kinematics problem is an important study in the robotic motion control. The mapping from joint space to Cartesian task space is referred to as direct kinematics and mapping from Cartesian task space to joint space is referred to as inverse kinematics [1]. Because of the complexity of inverse kinematics, it is usually more difficult than forward kinematics to find the solutions [2–5]. In addition, when robot manipulator executes a motion control, the complicated inverse kinematics computation consumes much CPU time and it certainty slows down the motion performance of robot manipulator. Therefore, solving this problem becomes an important issue.

For the progress of very large scale integration (VLSI) technology, the field programmable gate arrays (FPGAs) have been widely investigated due to their programmable hardwired feature, fast time to market, shorter design cycle, embedding processor, low power consumption, and higher density for the implementation of the digital system. FPGA provides a compromise between the special-purpose application specified integrated circuit (ASIC) hardware and general-purpose processors. Hence, many practical applications in industrial control [6], multiaxis motion control [7], and robotic control [8–10] have been studied. Therefore, for speeding up the computational power, the forward and inverse kinematics based on VHDL are studied in this paper. And the VHDL is applied to describe the overall behavior of the forward and inverse kinematics.

In recent years, an electronic design automation (EDA) simulator link, which can provide a cosimulation interface between MALTAB/Simulink [11] and HDL simulators-ModelSim [12], has been developed and applied in the design of the control system [13]. Using it, you can verify a VHDL, Verilog, or mixed-language implementation against your Simulink model or MATLAB algorithm. In MATLAB/Simulink environment, it can generate stimuli to ModelSim and analyze the simulation's responses [11]. In this paper, a cosimulation by EDA simulator link is applied to the proposed

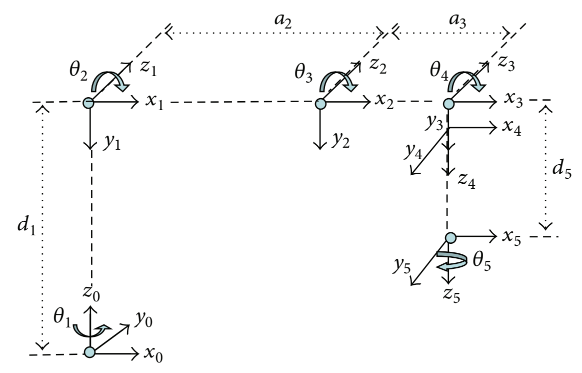

Figure 1: The link coordinates system of a five-axis articulated robot arm.

Table 1: Denavit-Hartenberg parameters for robot arm in Figure 1.

Link i	d_i (mm)	a_i (mm)	α_i	θ_i
1	$d_1 = 275$	0	$\alpha_1 = -\pi/2$	θ_1
2	0	$a_2 = 275$	0	θ_2
3	0	$a_3 = 255$	0	θ_3
4	0	0	$\alpha_4 = -\pi/2$	θ_4
5	$d_5 = 195$	0	0	θ_5

forward kinematics and inverse kinematics hardware. Some simulation results based on EDA simulator link will demonstrate the correctness and effectiveness of the forward and inverse kinematics.

2. Description of the Forward and Inverse Kinematics

A typical five-axis articulated robot arm is studied in this paper. Figure 1 shows its link coordinate system by Denavit-Hartenberg convention. Table 1 illustrates the values of the kinematics parameters. The forward kinematics of the articulated robot arm is the transformation of joint space R^5 $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ to Cartesian space R^3 (x, y, z). Conversely, the inverse kinematics of the articulated robot arm will transform the coordinates of robot manipulator from Cartesian space R^3 (x, y, z) to the joint space R^5 $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$. The computational procedure of forward and inverse kinematics is shown in Figure 1 and Table 1.

A coordinate frame is assigned to each link based on Denavit-Hartenberg notation. The transformation matrix for each link from frame i to $i-1$ is given by

$$
\begin{aligned}
{}^{i-1}A_i &= T(Z,d)\,T(Z,\theta)\,T(X,a)\,T(X,\alpha) \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}
\tag{1}
$$

where $T(Z,\theta)$ and $T(X,\alpha)$ present rotation and the $T(Z,d)$ and $T(X,\alpha)$ denote translation. Substituting the parameters in Table 1 into (1), the coordinate five matrixes respected with five axes of robot arm are shown as follows:

$$
\begin{aligned}
{}^{0}A_1 &= T(Z,d_1)\,T(Z,\theta_1)\,T(X,0)\,T\left(X,-\frac{\pi}{2}\right) \\
&= \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
{}^{1}A_2 &= T(Z,0)\,T(Z,\theta_2)\,T(X,a_2)\,T(X,0) \\
&= \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
{}^{2}A_3 &= T(Z,0)\,T(Z,\theta_3)\,T(X,a_3)\,T(X,0) \\
&= \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_3\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & a_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
{}^{3}A_4 &= T(Z,0)\,T(Z,\theta_4)\,T(X,0)\,T\left(X,-\frac{\pi}{2}\right) \\
&= \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & 0 \\ \sin\theta_4 & 0 & \cos\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
{}^{4}A_5 &= T(Z,d_5)\,T(Z,\theta_5)\,T(X,0)\,T(X,0) \\
&= \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
\tag{2}
$$

The forward kinematics of the end-effector with respect to the base frame is determined by multiplying five matrices from (2) as given above. An alternative representation of ${}^0A_5$ can be written as

$$\begin{aligned} {}^RT_H &= {}^0A_5 = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 \\ &\cdot {}^4A_5 \triangleq \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3}$$

The (n, o, a) are the orientation in the Cartesian coordinate system which is attached to the end-effector. Using the homogeneous transformation matrix to solve the kinematics problems, its transformation specifies the location (position and orientation) of the end-effector and the vector p presents the position of end-effector of robot arm. By multiplying five matrices and substituting into (3) and then comparing all the components of both sides after that we can solve the forward kinematics of the five-axis articulated robot arm as follows:

$$n_x = \cos\theta_1 \cos\theta_{234} \cos\theta_5 + \sin\theta_1 \sin\theta_5, \tag{4}$$

$$n_y = \sin\theta_1 \cos\theta_{234} \cos\theta_5 - \cos\theta_1 \sin\theta_5, \tag{5}$$

$$n_z = -\sin\theta_{234} \cos\theta_5, \tag{6}$$

$$o_x = -\cos\theta_1 \cos\theta_{234} \sin\theta_5 + \sin\theta_1 \cos\theta_5, \tag{7}$$

$$o_y = -\sin\theta_1 \cos\theta_{234} \sin\theta_5 - \cos\theta_1 \cos\theta_5, \tag{8}$$

$$o_z = \sin\theta_{234} \sin\theta_5, \tag{9}$$

$$a_x = -\cos\theta_1 \sin\theta_{234}, \tag{10}$$

$$a_y = -\sin\theta_1 \sin\theta_{234}, \tag{11}$$

$$a_z = -\cos\theta_{234}, \tag{12}$$

$$p_x = \cos\theta_1 \left(a_2 \cos\theta_2 + a_3 \cos\theta_{23} - d_5 \sin\theta_{234}\right), \tag{13}$$

$$p_y = \sin\theta_1 \left(a_2 \cos\theta_2 + a_3 \cos\theta_{23} - d_5 \sin\theta_{234}\right), \tag{14}$$

$$p_z = d_1 - a_2 \sin\theta_2 - a_3 \sin\theta_{23} - d_5 \cos\theta_{234}, \tag{15}$$

where

$$\theta_{23} = \theta_2 + \theta_3, \tag{16}$$

$$\theta_{234} = \theta_2 + \theta_3 + \theta_4. \tag{17}$$

The position vector p directs the location of the origin of the (n, o, a) frame which is defined to let the end-effector of robot arm by always gripping from a top down position. Therefore, the matrix in (3) is set by the following form:

$${}^RT_H = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & -1 & 0 & y \\ 0 & 0 & -1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{18}$$

Comparing the element (3,3) in (18) with (12), we obtained

$$\cos\theta_{234} = 1. \tag{19}$$

Therefore, we can get

$$\theta_{234} = 0. \tag{20}$$

Further, comparing the element (1,1) in (18) with (4), we obtained

$$\cos\left(\theta_1 - \theta_5\right) = 1. \tag{21}$$

Therefore, we can get

$$\theta_1 - \theta_5 = 0 \quad \text{or} \quad \theta_5 = \theta_1. \tag{22}$$

Let us assume that

$$b = a_2 \cos\theta_2 + a_3 \cos\theta_{23}; \tag{23}$$

then substituting (19)~(22) into (13)~(15), we can get the sequence for computations inverse kinematics as follows:

$$x = \cos\theta_1 \cdot b \tag{24}$$

$$y = \sin\theta_1 \cdot b \tag{25}$$

$$z = d_1 - a_2 \sin\theta_2 - a_3 \sin\theta_{23} - d_5. \tag{26}$$

From (24) and (25), we can get

$$\begin{aligned} b &= \pm\sqrt{\left(x^2 + y^2\right)}, \\ \theta_1 &= \theta_5 = a\tan 2\left(\frac{y}{x}\right). \end{aligned} \tag{27}$$

From (23) and (26), we can get

$$\theta_3 = \arccos\left(\frac{b^2 + \left(d_1 - d_5 - z\right)^2 - a_2^2 - a_3^2}{2a_2a_3}\right). \tag{28}$$

Once θ_3 is obtained, substitute it to (23) and (26) to get

$$\begin{aligned} b &= \left(a_2 + a_3 \cos\theta_3\right) \cos\theta_2 + a_3 \sin\theta_3 \sin\theta_2, \\ d_1 - d_5 - z &= \left(a_2 + a_3 \cos\theta_3\right) \sin\theta_2 + a_3 \sin\theta_3 \cos\theta_2. \end{aligned} \tag{29}$$

From (29), solve the linear equation in order to find the $\sin\theta_2$ and $\cos\theta_2$ as

$$\begin{aligned} \sin\theta_2 &= \frac{\left(a_2 + a_3 \cos\theta_3\right)\left(d_1 - d_5 - z\right) - a_3 b \sin\theta_3}{b^2 + \left(d_1 - d_5 - z\right)^2}, \\ \cos\theta_2 &= \frac{\left(a_2 + a_3 \cos\theta_3\right) b + a_3 \sin\theta_3 \left(d_1 - d_5 - z\right)}{b^2 + \left(d_1 - d_5 - z\right)^2}. \end{aligned} \tag{30}$$

Therefore, θ_2 can be derived as follows:

$$\theta_2 = a\tan 2\left[\frac{(a_2 + a_3\cos\theta_3)(d_1 - d_5 - z) - a_3 b\sin\theta_3}{(a_2 + a_3\cos\theta_3)b + a_3\sin\theta_3(d_1 - d_5 - z)}\right]. \quad (31)$$

Further, from (17) and (20), θ_4 is obtained as

$$\theta_4 = -\theta_2 - \theta_3. \quad (32)$$

Finally, the forward kinematics and inverse kinematics of the five-axis articulated robot arm, based on the assumption in (18) in which the end-effector of the robot arm is always toward the top down direction, can be summarized as follows.

For computing the forward kinematics, consider the following steps.

Step 1. Consider

$$\text{end}\,x = \cos\theta_1(a_2\cos\theta_2 + a_3\cos\theta_{23} - d_5\sin\theta_{234}). \quad (33)$$

Step 2. Consider

$$\text{end}\,y = \sin\theta_1(a_2\cos\theta_2 + a_3\cos\theta_{23} - d_5\sin\theta_{234}). \quad (34)$$

Step 3. Consider

$$\text{end}\,z = d_1 - a_2\sin\theta_2 - a_3\sin\theta_{23} - d_5\cos\theta_{234}. \quad (35)$$

In the previous steps, end x, end y, and end z are the position of end point which are the same as x, y, and z in (18).

For computing the inverse kinematics, consider the following steps.

Step 1. Consider

$$\theta_1 = \theta_5 = a\tan 2(\text{end}\,y, \text{end}\,x). \quad (36)$$

Step 2. Consider

$$b = \pm\sqrt{\text{end}\,x^2 + \text{end}\,y^2}. \quad (37)$$

Step 3. Consider

$$\theta_3 = a\cos\left(\frac{b^2 + (d_1 - d_5 - \text{end}\,z)^2 - a_2^2 - a_3^2}{2a_2a_3}\right). \quad (38)$$

Step 4. Consider

$$S_2 = (a_2 + a_3\cos\theta_3)(d_1 - d_5 - \text{end}\,z) - a_3 b\sin\theta_3. \quad (39)$$

Step 5. Consider

$$C_2 = (a_2 + a_3\cos\theta_3)b + a_3\sin\theta_3(d_1 - d_5 - \text{end}\,z). \quad (40)$$

Step 6. Consider

$$\theta_2 = a\tan 2(S_2, C_2). \quad (41)$$

Step 7. Consider

$$\theta_4 = -\theta_2 - \theta_3. \quad (42)$$

The parameters of robot arm a_2, a_3, d_1, d_5 are shown in Table 1.

3. Computations of Trigonometric Function and Its Hardware Implementation

Before performing the computation of the forward and inverse kinematics for five-axis articulated robot arm, some key trigonometric functions need to be built up as a component for being applied, and those are sine function, cosine function, arctangent function, and arccosine function. To increase the computing accuracy, LUT (look-up table) technique and Taylor series expanse technique are used to the computational algorithm design of the arctangent function and arccosine function. However, the computation algorithm used in these two functions is very similar; therefore, the detailed design methods, only sine/cosine function, and arctangent function are described as follows.

3.1. Computation Algorithm and Hardware Realization of Sine Function and Cosine Function. To compute $\sin(\theta)$ and $\cos(\theta)$ functions, the $\theta = \theta_I + \theta_F$ is firstly defined, in which θ_I and θ_F represented the integer part and fraction part of the θ, respectively. Then the formulation of these two functions is expanded as follows:

$$\begin{aligned}\sin(\theta_I + \theta_F) &= \sin(\theta_I)\cos(\theta_F) + \cos(\theta_I)\sin(\theta_F)\\ \cos(\theta_I + \theta_F) &= \cos(\theta_I)\cos(\theta_F) - \sin(\theta_I)\sin(\theta_F).\end{aligned} \quad (43)$$

In the hardware design, θ is adopted as 16-bit Q7 format. Therefore, if θ is 0000001001000000, it represents 4.5 degree. In addition, four LUTs (look-up tables) are built up to store the values of $\sin(\theta_I)$, $\cos(\theta_I)$, $\sin(\theta_F)$, and $\cos(\theta_F)$ functions. The LUT for $\sin(\theta_I)$ and $\cos(\theta_I)$ functions stores 360 pieces of data with 24-bit Q23 format and that for $\sin(\theta_F)$ and $\cos(\theta_F)$ functions stores 128 pieces of data with the same 24-bit Q23 format. Therefore, according to (43), the results of sine and cosine with 16-bit Q15 format can be computed after looking up four tables. In the realization, finite state machine (FSM) is adopted and the example to compute the cosine function is shown in Figure 2. It presents four LUTs, two multipliers, and one adder in hardware which manipulates 6 steps to complete the overall computation. Due to the fact that the operation of each step is 20 ns (50 MHz) in FPGA, a total of 6 steps only need 120 ns operation time. In addition, the FPGA (Altera Cyclone IV) resource usage for the realization of the sine or cosine function needs 232 logic elements (LEs) and 30,720 RAM bits.

3.2. Computation Algorithm and Hardware Realization of Arctangent Function. The equation of arctangent function is shown as follows:

$$\theta = a\tan 2\left(\frac{y}{x}\right), \quad (44)$$

where the inputs are x and y and the output is θ. Herein, there are two steps to evaluate the arctangent function.

(1) First Step. $\theta_1 = f(x) = \tan^{-1}(x)$ is computed by using Taylor series expansion and the input values are defined within $1 \le x \le 0$ (or the output value: $45^\circ \le \theta_1 \le 0^\circ$).

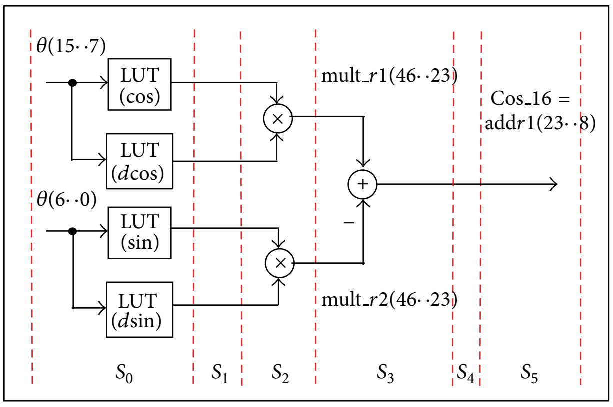

FIGURE 2: FSM for computing the *cosine* function.

The third-order polynomial is considered and the expression within the vicinity of x_0 is shown as follows:

$$f(x) \cong a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + a_3(x - x_0)^3, \tag{45}$$

with

$$\begin{aligned} a_0 &= f(x_0) = \tan^{-1}(x_0), \\ a_1 &= f'(x_0) = \frac{1}{1 + x_0^2}, \\ a_2 &= f''(x_0) = \frac{-2x_0}{(1 + x_0^2)^2}, \\ a_3 &= f^{(3)}(x_0) = \frac{-2 + 6x_0^2}{(1 + x_0^2)^3}. \end{aligned} \tag{46}$$

Actually, in realization, only third-order expansion in (44) is not enough to obtain an accuracy approximation due to the reason that the large error will occur when the input x is far from x_0. To solve this problem, combining the LUT technique and Taylor series expanse technique is considered. To set up the LUT, several specific values for x_0 within the range $1 \le x \le 0$ are firstly chosen; then the parameters from a_0 to a_3 in (46) are computed. Those data included x_0 and a_0 to a_3 will be stored to LUT. Following that, when it needs to compute $\tan^{-1}(x)$ in (45), the x_0 which is the most approximate to input x and its related a_0 to a_3 will be selected from LUT and then perform the computing task.

(2) Second Step. After completing the computation of $\tan^{-1}(x)$, we can evaluate $\theta = a\tan 2(y/x)$ further and let the output suitable to the range be within $-180^\circ \le \theta \le 180^\circ$. The formulation for each region in X-Y coordinate is shown in Figure 3.

In hardware implementation, the inputs and output of the arctangent function are designed with 32-bit Q15 and 16-bit Q15 format, respectively. It consists of one main circuit with FSM architecture and two components for computing the divider and $\tan^{-1}(x)$ functions. However, the design and implementation of the $\tan^{-1}(x)$ function is a major task. The input and output values of component $\tan^{-1}(x)$ all belong

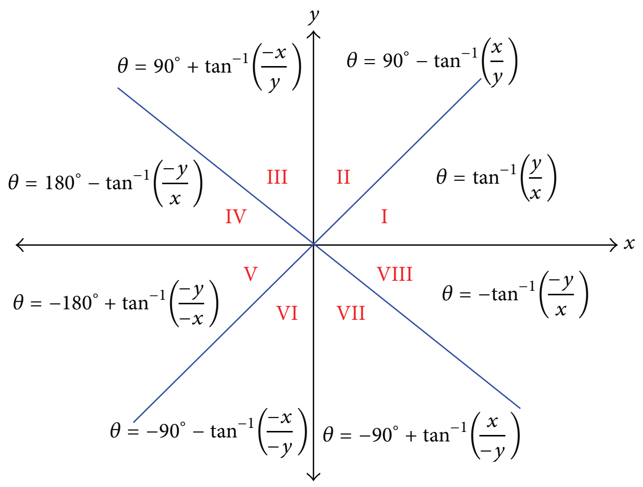

FIGURE 3: Compute $\theta = a\tan 2(y/x)$ of each region in X-Y coordinates.

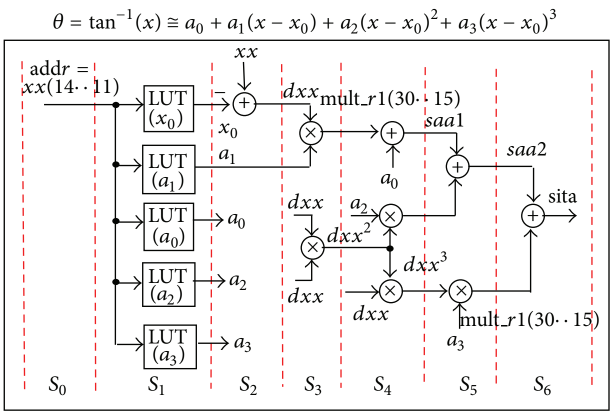

FIGURE 4: FSM to compute $\tan^{-1}(x)$ function.

to 16-bit Q15 format. The FSM is employed to model the computation of $\tan^{-1}(x)$ and is shown in Figure 4 which uses two multipliers and one adder in the design. The multiplier and adder apply Altera LPM (library parameterized modules) standard. In Figure 4, it manipulates 7-step machine to carry out the overall computations of $\tan^{-1}(x)$. The steps $s_0 \sim s_1$ execute to look up 5 tables and $s_2 \sim s_6$ perform the computation of polynomial in (45). Further, according to the computation logic shown in Figure 3, it uses 16 steps to complete the $a\tan 2(y/x)$ function. Due to the fact that the operation of each step is 20 ns (50 MHz) in FPGA, a total of 16 steps only need 320 ns operation time. In addition, the FPGA (Altera Cyclone IV) resource usage for the realization of the arctangent function needs 4,143 LEs and 1,280 RAM bits.

4. Design and Hardware Implementation of Forward/Inverse Kinematics and Its Simulation Results

The block diagram of kinematics for five-axis articulated robot arm is shown in Figure 5. The inputs are the end-point position by end x, end y, and end z which is relative

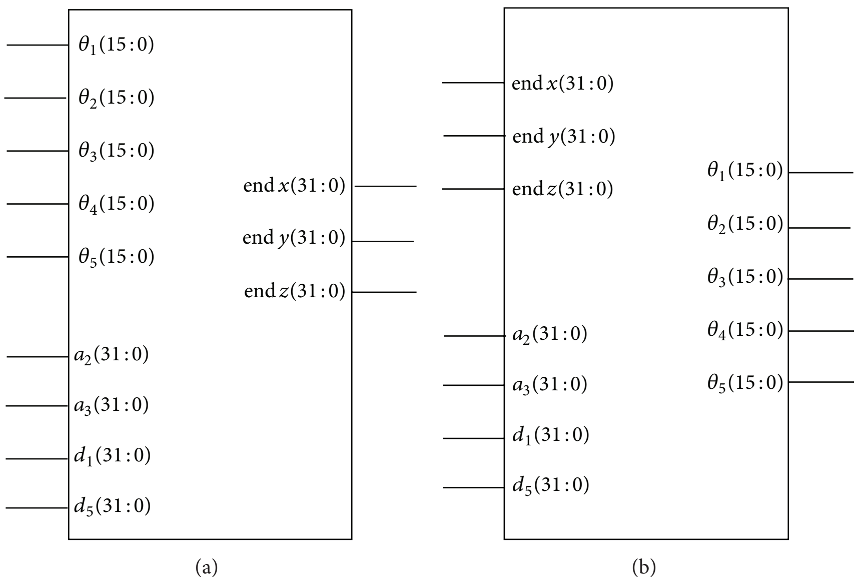

FIGURE 5: Block diagram for (a) forward kinematics and (b) inverse kinematics.

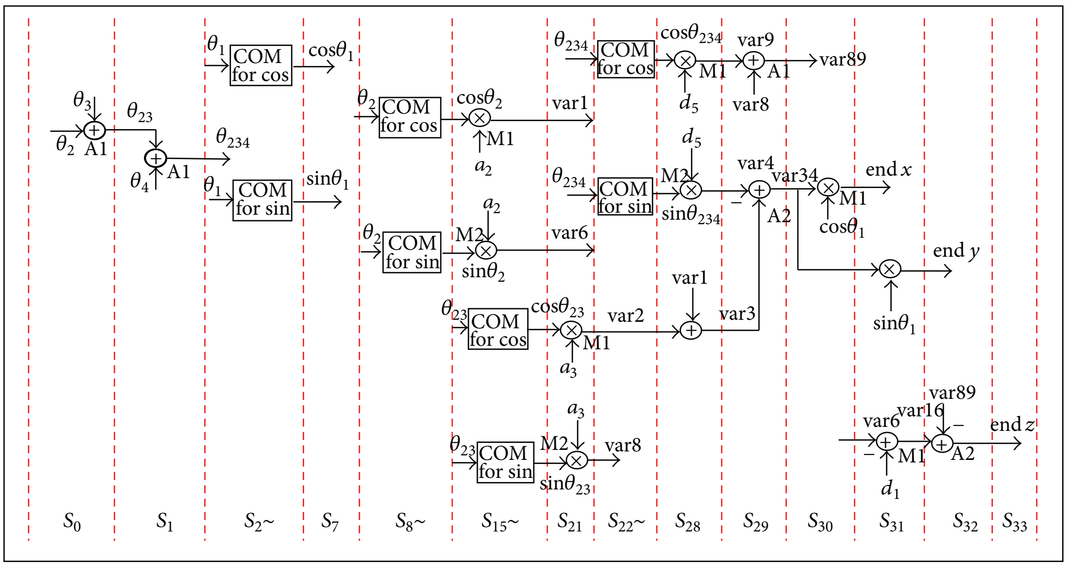

FIGURE 6: FSM for computing the forward kinematics.

to (x, y, z), as well as the mechanical parameters by a_2, a_3, d_1, and d_5. The outputs are mechanical angles $\theta_1 \sim \theta_5$. In Figure 5, the parameters of mechanical length and the end-point position are designed with 32-bit Q15 data format and the parameters of mechanical angle are designed with 16-bit Q7 data format. According to the formulations of forward and inverse kinematics which are described in Section 2, the finite state machine (FSM) method is applied to design the hardware for reducing the usage of hardware resource. Herein, the FSM designs to compute the forward kinematics and inverse kinematics are, respectively, shown in Figures 6 and 7. The FSM will generate sequential signals and will step-by-step compute the forward kinematics and inverse kinematics. The implementation of inverse kinematics is developed by VHDL. In Figure 6, there are 34 steps to present the computations of forward kinematics, and the circuit includes two multipliers, two adders, one component for sine function, and one component for cosine function. In Figure 7, there are 47 steps to perform the inverse kinematics, and the circuit needs three multipliers, two dividers, two adders, one square root function, one component for arctangent function, one component for arccosine function, one component for sine function, and one component for cosine function. The notation "COM" in Figures 6~7 is represented with "component." For example, the "COM for cos" is the component of cosine function. The designs of the component regarded as trigonometric function are described in previous section. Due to the fact that the operation of each step is 20 ns (50 MHz) in FPGA,

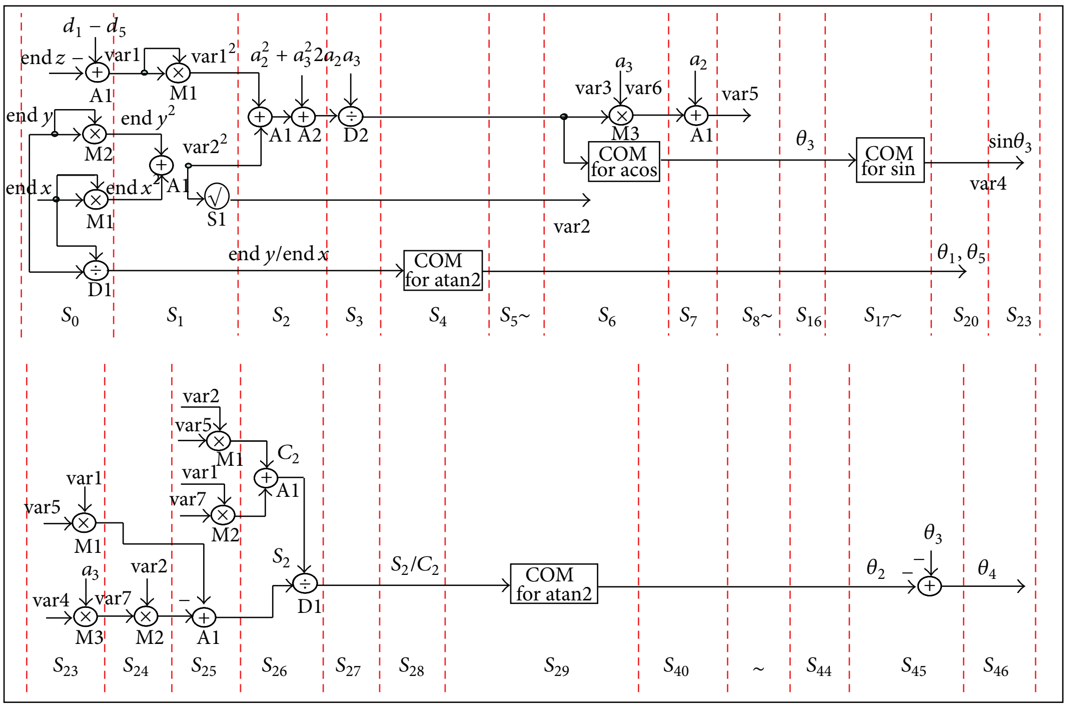

FIGURE 7: FSM for computing the inverse kinematics.

the executing time for the computation of forward and inverse kinematics is 680 ns and 940 ns, respectively. In addition, the FPGA (Altera Cyclone IV) resource usage for the realization of the forward kinematics IP and the inverse kinematics IP is 1,575 LEs and 30,720 RAM bits and 9,400 LEs and 84,224 RAM bits, respectively.

The cosimulation architecture by ModelSim/SimuLink for the proposed forward kinematics is shown in Figure 8 and that for the inverse kinematics is shown in Figure 9, whose works in ModelSim execute the function of the computing the forward and inverse kinematics. The input values to ModelSim are provided from Simulink and, through the computation working in ModelSim, output responses are displayed to Simulink. To confirm the correctness, an embedded Matlab function for computing the forward and inverse kinematics is considered. Under the cosimulation of ModelSim and Simulink architecture, the simulation results are presented as follows.

Mechanical parameters of the five-axis articulated robot arm are chosen by

$$a_2 = 275, \qquad a_3 = 255, \qquad d_1 = 275, \qquad d_5 = 195 \text{ (mm)}. \tag{47}$$

Simulation results of inverse kinematics with the following two cases are listed in Table 2. The inputs of inverse kinematics are the end-effectors of robot arm presented by end x, end y, and end z (mm) and the outputs are five joint angles presented by $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ (degree).

In the first case, the inputs end x, end y, and end z are chosen as

$$\text{end}\,x = 300, \qquad \text{end}\,y = 299, \qquad \text{end}\,z = -150. \tag{48}$$

In the second case, the inputs end x, end y, and end z are chosen as

$$\text{end}\,x = 60, \qquad \text{end}\,y = 180, \qquad \text{end}\,z = 450. \tag{49}$$

Simulation results of forward kinematics with the following two cases are listed in Table 3. The inputs of forward kinematics are five joint angles presented by $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ (degree) and the outputs are end-effectors presented by end x, end y, and end z (mm).

In the first case, the inputs $\theta_1, \theta_2, \theta_3, \theta_4$, and θ_5 are chosen as

$$\theta_1 = 44.9043477, \qquad \theta_2 = 4.8948685, \qquad \theta_3 = 49.1952721, \qquad \theta_4 = -54.0901472, \qquad \theta_5 = 44.9043477. \tag{50}$$

In the second case, the inputs $\theta_1, \theta_2, \theta_3, \theta_4$, and θ_5 are chosen as

$$\theta_1 = 71.5650511, \qquad \theta_2 = -99.4939347, \qquad \theta_3 = 76.703125, \qquad \theta_4 = 22.7812500, \qquad \theta_5 = 71.5650511. \tag{51}$$

From these two cases, the absolute difference of end-effectors end x, end y, and end z which are calculated from ModelSim and from Matlab is less than 0.05 mm, and θ_1 to θ_5 are less than 0.01°. It shows that the proposed forward and inverse kinematics for the five-axis articulated robot arm are correct and effective.

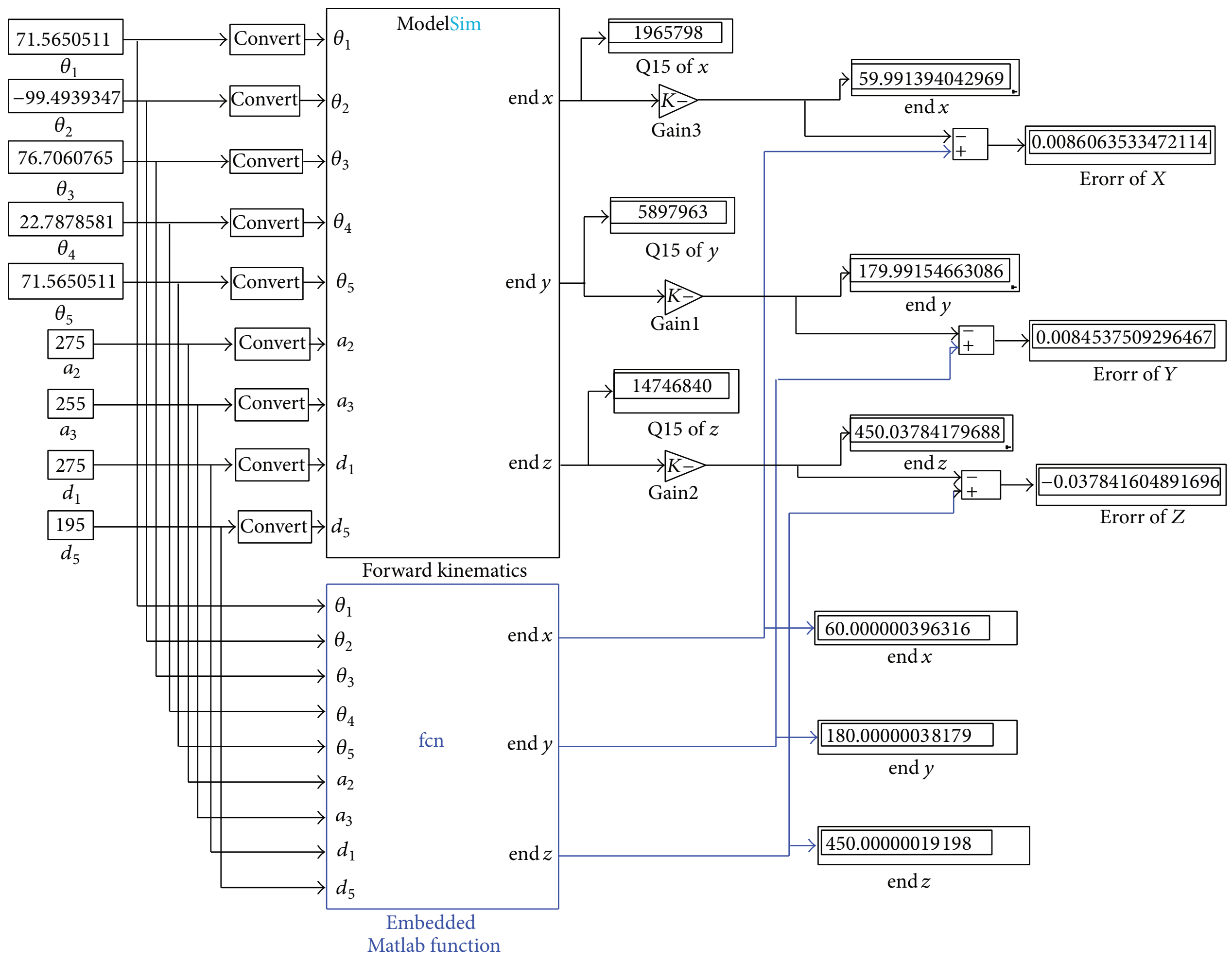

Figure 8: Cosimulation architecture of forward kinematics using ModelSim and SimuLink.

Table 2: Two cases' simulation results of inverse kinematics from ModelSim and Matlab.

Output	Input					
	The 1st case (end x, end y, end z)(1)			The 2nd case (end x, end y, end z)(2)		
	ModelSim	Matlab	Error	ModelSim	Matlab	Error
θ_1	44.8984375	44.9043477	0.0059702	71.5703125	71.5650511	−0.0052613
θ_2	4.8906250	4.8948685	0.0042435	−99.4843750	−99.4939347	0.0095597
θ_3	49.1953125	49.1952721	−0.0000400	76.7060765	76.7031250	0.0029515
θ_4	−54.0859375	−54.0901472	0.0042032	22.7878581	22.7812500	0.0066081
θ_5	44.8984375	44.9043477	0.0059102	71.5703125	71.5650511	0.0052613

(1) denotes the case 1 shown in (48).
(2) denotes the case 2 shown in (49).

Table 3: Two cases' simulation results of forward kinematics from ModelSim and Matlab.

Output	Input					
	The 1st case ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$)(1)			The 2nd case ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$)(2)		
	ModelSim	Matlab	Error	ModelSim	Matlab	Error
end x	300.0321650	300.0000160	−0.0321490	59.9913940	60.0000000	0.0086063
end y	299.0496215	299.0000160	−0.0496054	179.9915466	180.0000000	0.0084537
end z	−149.9700920	−149.9999999	−0.0299906	450.0378417	450.0000000	−0.0378416

(1) denotes the case 1 shown in (50).
(2) denotes the case 2 shown in (51).

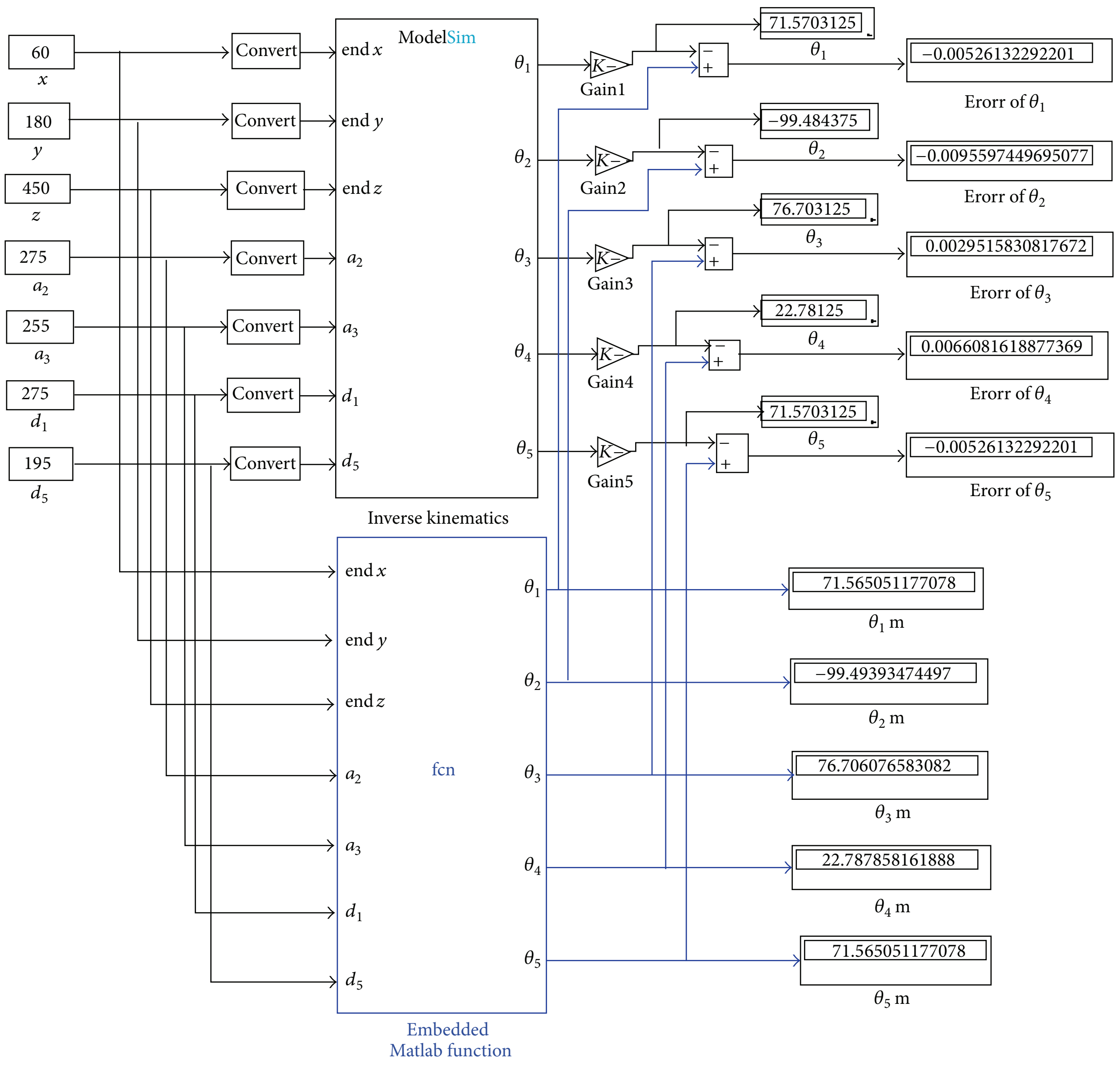

FIGURE 9: Cosimulation architecture of inverse kinematics using ModelSim and SimuLink.

5. Conclusions

The forward kinematics and inverse kinematics of five-axis articulated robot arm, in which the end-effector of the robot arm is always toward the top down direction, have been successfully demonstrated in this paper. Through the cosimulation of ModelSim and Simulink, the accuracy under two examples of forward and inverse kinematics with the results of error of the end-effectors end x, end y, and end z is less than 0.05 mm and the error for $\theta_1 \sim \theta_5$ is less than 0.01°. Further, the executing times for the computations of forward and inverse kinematics in FPGA are only 680 ns and 940 ns, respectively. The high speed computational power and reasonable accuracy apparently increase the motion performance of the five-axis articulated robot arm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by National Science Council of the R.O.C. under Grant no. NSC 100-2632-E-218 -001-MY3.

References

[1] J. J. Craig, *Introduction to Robotics, Mechanics & Control*, Addison-Wesley, New York, NY, USA, 1986.

[2] W. Shen, J. Gu, and E. E. Milios, "Self-configuration fuzzy system for inverse kinematics of robot manipulators," in *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '06)*, pp. 41–45, June 2006.

[3] P. Falco and C. Natale, "On the stability of closed-loop inverse kinematics algorithms for redundant robots," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 780–784, 2011.

[4] S.-W. Park and J.-H. Oh, "Hardware realization of inverse kinematics for robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 1, pp. 45–50, 1994.

[5] G.-S. Huang, C.-K. Tung, H.-C. Lin, and S.-H. Hsiao, "Inverse kinematics analysis trajectory planning for a robot arm," in *Proceedings of the 8th Asian Control Conference (ASCC '11)*, pp. 965–970, twn, May 2011.

[6] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 224–243, 2011.

[7] J. U. Cho, Q. N. Le, and J. W. Jeon, "An FPGA-based multiple-axis motion control chip," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 856–870, 2009.

[8] Y.-S. Kung, K.-H. Tseng, C.-S. Chen, H.-Z. Sze, and A.-P. Wang, "FPGA-implementation of inverse kinematics and servo controller for robot manipulator," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '06)*, pp. 1163–1168, December 2006.

[9] S. Sánchez-Solano, A. J. Cabrera, I. Baturone, F. J. Moreno-Velo, and M. Brox, "FPGA implementation of embedded fuzzy controllers for robotic applications," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 4, pp. 1937–1945, 2007.

[10] C. C. Wong and C. C. Liu, "FPGA realisation of inverse kinematics for biped robot based on CORDIC," *Electronics Letters*, vol. 49, no. 5, pp. 332–334, 2013.

[11] The Mathworks, *Matlab/Simulink Users Guide*, Application Program Interface Guide, 2004.

[12] Modeltech, *ModelSim Reference Manual*, 2004.

[13] Y.-S. Kung, N. V. Quynh, C.-C. Huang, and L.-C. Huang, "Simulink/ModelSim co-simulation of sensorless PMSM speed controller," in *Proceedings of the IEEE Symposium on Industrial Electronics and Applications (ISIEA '11)*, pp. 24–29, September 2011.