Hindawi
Journal of Robotics
Volume 2023, Article ID 2426982, 8 pages
https://doi.org/10.1155/2023/2426982

Research Article

The Parallel Solving Method of Robot Kinematic Equations Based on FPGA

Deli Zhang,[1] **Shaohua Jiang**,[1] **and Liu Zhe**[2]

[1]*College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*
[2]*Engineering Technology Center, Shenyang Aircraft Company, Shenyang 110850, China*

Correspondence should be addressed to Deli Zhang; nuaazdl@126.com

Received 13 May 2023; Revised 6 October 2023; Accepted 10 October 2023; Published 3 November 2023

Academic Editor: Shahram Payandeh

In the implementation of robot motion control, complex kinematic computations consume too much central processing unit (CPU) time and affect the responsiveness of robot motion. To solve this problem, this paper proposes a parallel method for solving kinematic equations of articulated robots based on the coordinate rotation digital computer (CORDIC) algorithm. The method completes the fast calculation of the transcendental function based on the CORDIC algorithm, adopts the tree structure method to optimize the key computational paths of forward and inverse solutions, and designs a parallel pipeline to realize the low latency and high throughput of the kinematic equations. The experiments of the proposed method are validated based on the field-programmable gate array (FPGA) hardware experimental platform, and the experimental results demonstrate that the computational time to complete the entire kinematic equations is 4.68 μs, of which the computational time for the kinematic positive solution is 0.52 μs and the computational time for the kinematic inverse solution is 4.16 μs.

1. Introduction

Real-time control of robots is a challenging research priority, especially in space, medical, and industrial robotics applications where fast response is very important [1, 2]. However, robot kinematics involves real-time computation of a large number of transcendental functions such as cosine, sine, arc tangent, square, and so on. Solving kinematic equations takes up a lot of CPU time, which makes it difficult for the robot to respond quickly [3–5].

To solve this problem, using digital signal processing (DSP) and field-programmable gate array (FPGA) are two possible solutions. Some researchers perform kinematic calculations for robots based on DSP [6–8], but as robot systems become increasingly complex, the computational tasks undertaken by DSP also become more intensive, and excessive resource occupation leads to a decrease in the robot's response speed. FPGA is widely used in the field of robot control due to its programmable hardwired characteristics and fast parallel computing capabilities, which improve the processing power of hardware and the speed of real-time information processing [9–12], Chand et al. [13] used embedded FPGA for motion planning and control of dual arm robots, by establishing accurate arm motion sequences to accurately execute multiple tasks. Gürsoy and Efe [14] proposed proportion integral differential and sliding mode control scheme for robot manipulators based on FPGA, achieving better trajectory tracking performance. Furthermore, kinematic solution based on FPGA has also been proposed by researchers.

The issue of the first importance is the computation of transcendental functions by FPGA, the look-up-table (LUT) method [15], Taylor series expansion method [16], and coordinate rotation digital computer (CORDIC) algorithm [17–19] have been proposed. Zhang et al. [15] proposed a master–slave surgical robot forward–reverse kinematics computation method based on FPGA, and all the transcendental function computations use the LUT method, which can effectively improve the computation rate of the transcendental function, but it needs to take up a large amount of LUT resources when performing high-precision computations. In order to solve the problem of kinematic inverse tangent and inverse cosine hardware computation, Kung et al. [16] proposed a combination of Taylor series expansion method and LUT method, which reduces the LUT resource occupation for high-precision computation, but requires multipliers for polynomial computation, which leads to a reduction in the computation rate. The CORDIC algorithm has high speed and area achieved in digital signal processing applications [20], Multiplexers based CORDIC algorithm and fully pipelined CORDIC algorithm [21] used to achieve a fast and efficient hardware on FPGA. Wei et al. [17], Zhang et al. [18], and Çelik et al. [19] have used the CORDIC algorithm for the computation of the transcendental function, which has a higher computation rate and less resource consumption when implemented in FPGA. In order to improve the calculation speed of kinematic equations, Zhang et al. [18] and Petko et al. [22] proposed an FPGA heterogeneous scheme for kinematic computation, where only the FPGA is used as a coprocessor for the fast computation of the transcendental function, which reduces the computation time consuming of the transcendental function, but increases the corresponding instruction scheduling time and puts forward higher requirements for the timing control of the heterogeneous platform. A single FPGA-based method for solving kinematic equations can effectively avoid the shortcomings of heterogeneous platform methods, with a simpler system composition and better robustness and stability. Chen et al. [23] proposed an FPGA-based kinematic intellectual property (IP) for selective compliance assembly robot arm (SCARA) robots, which realizes the overall kinematic computation within 10 μs, but due to the use of finite-state machines, this kinematic IP cannot perform parallel streaming computation and fails to give full play to the FPGA's parallel data processing capability. Fan et al. [24] proposed a high-level synthesis method based on Zynq FPGA to realize the inverse kinematics computation of humanoid robots, although the high-level synthesis method can realize the rapid development of FPGA, but it is more suitable for the scenarios with lower requirements on timing, the robot kinematics computation process is more complex, and the addition of the computation process to the timing control is more conducive to improving the real-time performance of the robot.

Motivated by the aforementioned discussions, an FPGA-based hardware parallel solving method for robot kinematic equations is proposed in this paper, which is based on the CORDIC algorithm to complete the fast computation of transcendental functions, adopts the tree structure method to optimize the key computational paths of the forward and inverse solutions, and carries out a parallel pipeline design of the whole forward and inverse solution computation to realize the low-latency and high-throughput solving of the kinematic equations. The main contributions and innovations of this paper are as follows:

(1) The kinematic hardware computation model proposed in this paper takes only 4.68 μs to complete the whole kinematic computation, and the computation process adopts parallel pipeline design, which allows data computation in each clock cycle and improves the computational efficiency greatly.

(2) The computational timing of the kinematic hardware computation model proposed in this paper is fixed, so that when the data bit width is increased, the computational cycle can still be guaranteed to remain unchanged, but the computational accuracy will be improved accordingly.

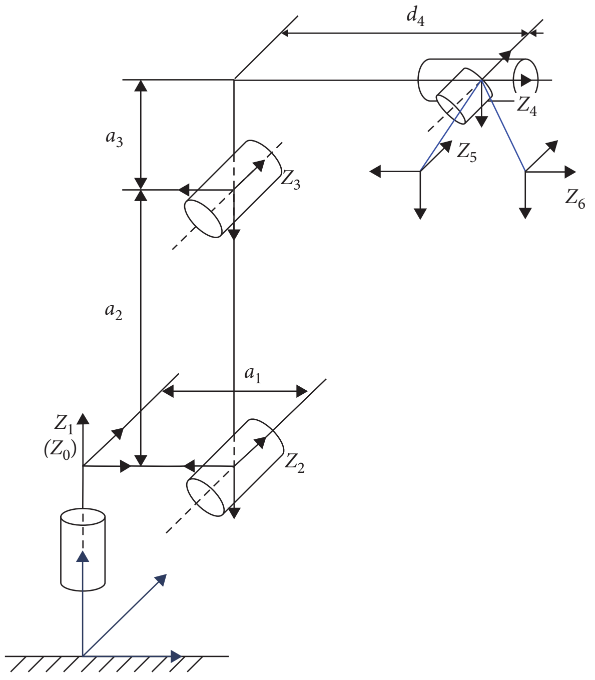


Figure 1: Link coordinate system of the robot.

The rest of the paper is organized as follows: in Section 2, the kinematic model is developed and forward/inverse kinematic equations are derived. The principle of CORDIC algorithm is presented in Section 3. In Section 4, a hardware parallel FPGA-based solution of the kinematic equations is given. Experimental results are given in Section 5. Conclusions are given in Section 6.

2. The Kinematic Equation for Articulated Robots

In this section, the structure of the robot is first described, then the robot is modeled using the Denavit–Hartenberg (D–H) method [25], and finally the forward kinematics equations and inverse kinematics equations are derived separately.

2.1. The Kinematic Model of Articulated Robots. An articulated robot is an open chain structure composed of a series of connecting linkages connected by joints [26, 27]. To accurately describe the robot's poses, this paper uses D–H method, which constructs a kinematic model using a fourth-order transformation homogeneous matrix to describe the relationships of adjacent links and deriving the positional relationships of each linkage relative to the base in a recursive manner.

The robot studied in this paper is a 6-degree-of-freedom industrial robot, and the kinematic model is built using D–H mothod, the link coordinate systemas$\{z_i\}$is shown in Figure 1, a_i is link length and d_i is link offset.

Table 1: Parameter table of the D–H method.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	0
2	$-\pi/2$	a_1	0	0
3	0	a_2	0	0
4	$-\pi/2$	a_3	d_4	0
5	$\pi/2$	0	0	0
6	$-\pi/2$	0	0	0

Based on the distribution of the coordinate system in Figure 1 and the connecting rod parameters, the D–H parameters to build this robot model can be derived, α_i is the link twist, α_i is link length, d_i is link offset, θ_i is the joint angle, as shown in Table 1.

In the D–H parameter method, the vector defined by the adjacent coordinate system $\{z_i\}$ is transformed to the coordinate system $\{z_{i-1}\}$, transformation for which homogeneous matrix can be expressed as [28]:

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where $c\theta_i$ means $\cos\theta_i$ and $s\theta_i$ stands for $\sin\theta_i$.

The D–H parameters in Table 1 are brought into Equation (1) to obtain the homogeneous transformation matrix between the adjacent linkages of this robot.

2.2. The Forward Kinematic Equations of Articulated Robots. After determining the linear transformation homogeneous matrix between each coordinate system, the kinematic equation ${}^0_6T$ can be obtained by multiplying the transformation matrix ${}^{i-1}_{\ \ i}T$ of each linkage:

$$ {}^0_6T = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The forward kinematic equation solution is to find the position ${}^0_6T$ of the end effector with respect to the reference coordinate system, knowing the parameters $(\alpha_i, a_i, d_i, \theta_i)$ of each joint. The inverse kinematic equation solution is to find the motion parameters θ_i of each joint based on the given poses ${}^0_6T$ of the end effector relative to the reference coordinate system.

2.3. The Inverse Kinematic Equations of Articulated Robots. The robot studied in this paper conforms to the Pieper criterion [29], i.e., three adjacent joint axes intersect at a point, and is a configuration with closed solutions. To meet the requirements of real-time robot control, this paper utilizes the algebraic method in the closed solution method to calculate the inverse kinematic solution, and the analytical expression can be obtained as:

$$ ikineSolver({}^0_6T) = (\theta_1, \theta_2, \ldots, \theta_6)^T \quad (3)$$

where $\theta_1 = \arctan(p_y, p_x) - \arctan(0, \pm\sqrt{p_x{}^2 + p_y{}^2})$,

$\theta_3 = \arctan(a_3, d_4) - \arctan(K, \pm\sqrt{1 - K^2})$,

$K = \frac{p_x{}^2 + p_y{}^2 + p_z{}^2 - 2a_1c_1p_x - 2a_1s_1p_y + a_1{}^2 - a_2{}^2 - a_3{}^2 - d_4{}^2}{2a_2\sqrt{a_3{}^2 + d_4{}^2}}$,

$\theta_2 = \theta_{23} - \theta_3$,

$\theta_{23} = \arctan(s_{23}, c_{23})$,

$s_{23} = (-a_2c_3 - a_3)p_z + (c_1p_x + s_1p_y - a_1)(a_2s_3 - d_4)$,

$c_{23} = (a_2s_3 - d_4)p_z + (c_1p_x + s_1p_y - a_1)(a_2c_3 + a_3)$,

$\theta_4 = \arctan(-a_xs_1 + a_yc_1, -a_xc_1c_{23} - a_ys_1c_{23} + a_zs_{23})$,

$\theta_5 = \arctan(s_5, c_5)$,

$s_5 = a_x(-c_1c_{23}c_4 - s_1s_4) - a_y(s_1c_{23}c_4 - c_1s_4) + a_z(s_{23}c_4)$,

$c_5 = a_x(-c_1s_{23}) + a_y(-s_1s_{23}) + a_z(-c_{23})$,

$\theta_6 = \arctan(s_6, c_6)$,

$s_6 = -n_x(c_1c_{23}s_4 - s_1c_4) - n_y(s_1c_{23}s_4 + c_1c_4) + n_z(s_{23}s_4)$,

$c_6 = n_x[(c_1c_{23}c_4 + s_1s_4)c_5 - c_1s_{23}s_5] + n_y[(s_1c_{23}c_4 - c_1s_4)c_5 - s_1s_{23}s_5] - n_z(s_{23}c_4c_5 + c_{23}s_5)$.

3. Parallel Calculation Method of Transcendental Functions for Kinematic Equations

The CORDIC algorithm is an iterative algorithm that uses only shift operations and addition and subtraction operations to solve the problem of real-time computation of trigonometric functions in air navigation control systems [30]. Based on this, Walther [31] proposed a unified form of the CORDIC algorithm, application for which extends to inverse trigonometric functions, hyperbolic functions, and transcendental functions. The algorithm is well-suited to run in platforms such as FPGAs due to its high hardware efficiency. From the perspective of FPGA portability, using fixed-point numbers for cordic algorithm implementation can ensure computational speed while configuring FPGA resource utilization. Therefore, this paper employs the CORDIC algorithm in the form of fixed-point numbers for trigonometric functions, inverse trigonometric functions, and open-root operations in the kinematic solution process.

The unified iterative equations of CORDIC algorithms for circular, linear, and hyperbolic systems are given as follows:

$$\begin{cases} x_{i+1} = x_i - \mu d_i\,(2^{-i}y_i) \\ y_{i+1} = y_i + d_i\,(2^{-i}x_i) \\ z_{i+1} = z_i - d_i\theta_i \end{cases} \quad (4)$$

where the circular system is $\mu = 1, \theta_i = \tan^{-1} 2^{-i}$, the linear system is $\mu = 0, \theta_i = 2^{-i}$, and the hyperbolic system is $\mu = -1, \theta_i = \tan h^{-1} 2^{-i}$.

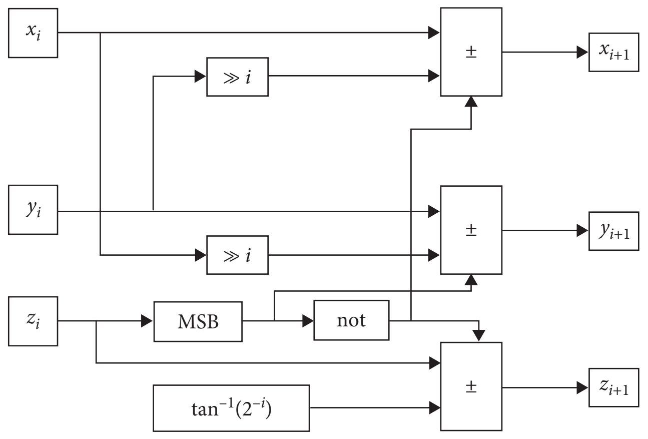


FIGURE 2: Iterative computing unit for rotation mode in CORDIC algorithm.

The CORDIC algorithm solves the computation of trigonometric and inverse trigonometric functions under the circumferential system, which contains the rotation mode and the vector mode. The former solves the computation of trigonometric functions, and the latter is for inverse trigonometric functions. In the calculation of Equation (3), there exist open-root operations in the form of $\sqrt{1-K^2}$. The CORDIC algorithm can be computed in the vector mode of hyperbolic systems.

The coefficients of the iterative Equation (4) of the CORDIC algorithm are different when performing trigonometric functions, inverse trigonometric functions, and open-root operations, but the implementation principle is the same. Taking the rotation mode of the circular system as an example, the operation unit of the iterative Equation (4) includes three adders, LUT, and two shift operations. The structure of the iterative processing unit for FPGA implementation is shown in Figure 2, $\{x_i y_i z_i\}$ is the input parameters of the iterative computing unit, $\{x_{i+1} y_{i+1} z_{i+1}\}$is the output parameter of the iterative computing unit.

The computational accuracy of CORDIC algorithms is determined by the iteration number. To ensure the computational accuracy of kinematics, the number of iterations is chosen to be 16, and the data bit width of 24 bit is selected for fixed-point computation in this paper. In order to maximize the computational efficiency of CORDIC algorithm, this paper adopts a parallel pipeline structure based on FPGA, and the hardware structure is shown in Figure 3; $\{x_0 y_0 z_0\}$ is the input parameter of the first iteration of the CORDIC algorithm and $\{x_{n-1} y_{n-1} z_{n-1}\}$is the output parameter of the Nth iteration. Certainly, the number of iterations can be set according to different accuracy requirements, thus balancing computational accuracy and resource consumption.

4. Hardware Parallel Solution Method for Kinematic Equations

On the basis of implementing the calculation of transcendental functions based on FPGA, this section carries out the parallel pipeline design of the computation process according to Equations (2) and (3). Finally gives the parallel computation hardware structure for the forward kinematics equation and the inverse kinematics equation, respectively.

4.1. Hardware Parallel Solution Method for the Forward Kinematic Solution. In the computation process of the kinematic forward solution, the logical path with the longest delay from the input to the output is the critical path, and the optimization degree of the key path determines the working speed of the model. In Equation (2), n_x, n_y, o_x, and o_y are the critical paths to calculate the forward kinematic solution, where $n_x = c_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] + s_1(s_4c_5c_6 + c_4s_6)$, c_i means $\cos\theta_i$, s_i means $\sin\theta_i$, c_{ij} means $\cos(\theta_i+\theta_j)$, s_{ij} means $\sin(\theta_i+\theta_j)$. The computational modules can be divided into different computational modules according to the order of operations: $A \times B$, $(A \pm D) \times B$, and $(A+D) \times B \pm C$, and each computational module is executed in parallel in different time sequences. In this paper, the nodes are computed in parallel through the tree structure method. The entire critical path execution process takes 21 clock cycles (*clk*) and requires 11-bit wide hard-core multipliers. The computational model of the kinematic forward solution n_x is shown in Figure 4, where θ_i is the joint angle, c_i means $\cos\theta_i$, s_i means $\sin\theta_i$, c_{ij} means $\cos(\theta_i+\theta_j)$, s_{ij} means $\sin(\theta_i+\theta_j)$, and $r_{i.j}$ is procedure calculated value.

Since each *clk* performs homotypic structural computation, the whole computation process can be parallel pipelined by inserting flow registers. After completing the structure and timing design of the critical path, it is necessary to perform a register leveling process for other computational paths so as to balance different computational paths and thus increase the overall working frequency. Since the critical path has already achieved the lowest latency output, the parallel pipeline solution of the forward kinematic solution is completed by inserting registers in other shorter computational paths, and inserting a pipeline register after dividing according to the same computational structure and the same computational timing. Due to the insertion of pipeline registers into the computational structure, the computation time of the forward kinematic positive solution increases to 26 *clk*, but the data throughput rate is multiplied.

4.2. Hardware Parallel Computation Method for the Inverse Kinematic Solution. In Equation (3), the joint angle θ_1 and θ_3 can be calculated as follows:

First: $\theta_1 = \arctan(p_y, p_x) - \arctan\left(0, \pm\sqrt{{p_x}^2 + {p_y}^2}\right)$

Second: $K = \dfrac{{p_x}^2+{p_y}^2+{p_z}^2-2a_1c_1p_x-2a_1s_1p_y+{a_1}^2-{a_2}^2-{a_3}^2-{d_4}^2}{2a_2\sqrt{{a_3}^2+{d_4}^2}}$

Third: $\theta_3 = \arctan(a_3, d_4) - \arctan\left(K, \pm\sqrt{1-K^2}\right)$

The process of solving the joint angle θ_1 and θ_3 involves trigonometric function, inverse trigonometric function, and open-root calculation, where the computation of process data K is the most time-consuming, and its computation process includes 16 multiplications, one open root, and one division. Besides, the computation process of joint angle is serial. In order to improve the computational efficiency of the kinematic inverse solution, the invariant constants in the computational process are precomputed. The computational

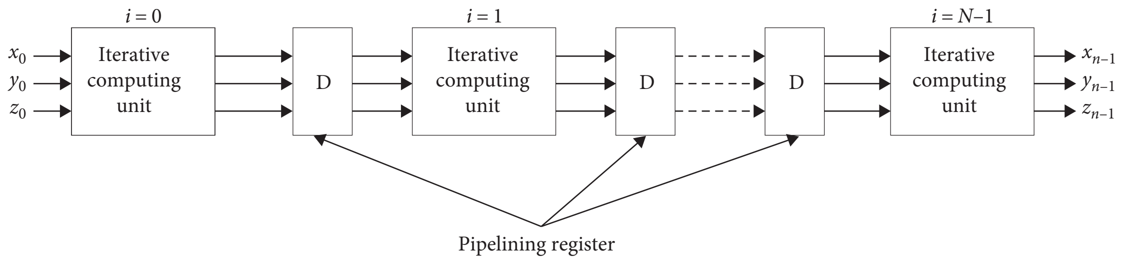


FIGURE 3: Hardware structure of parallel pipelining CORDIC algorithm.

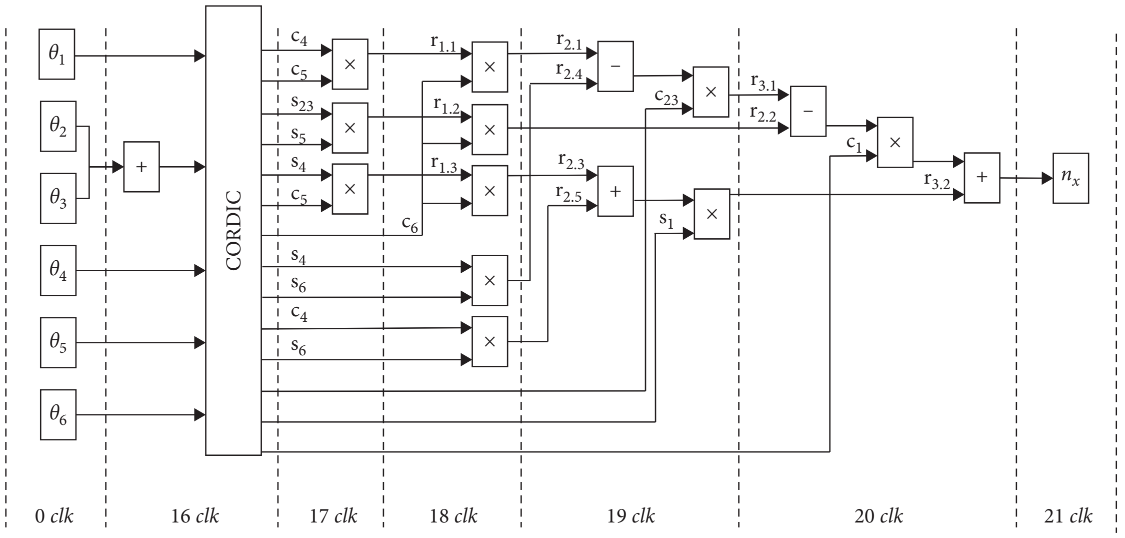
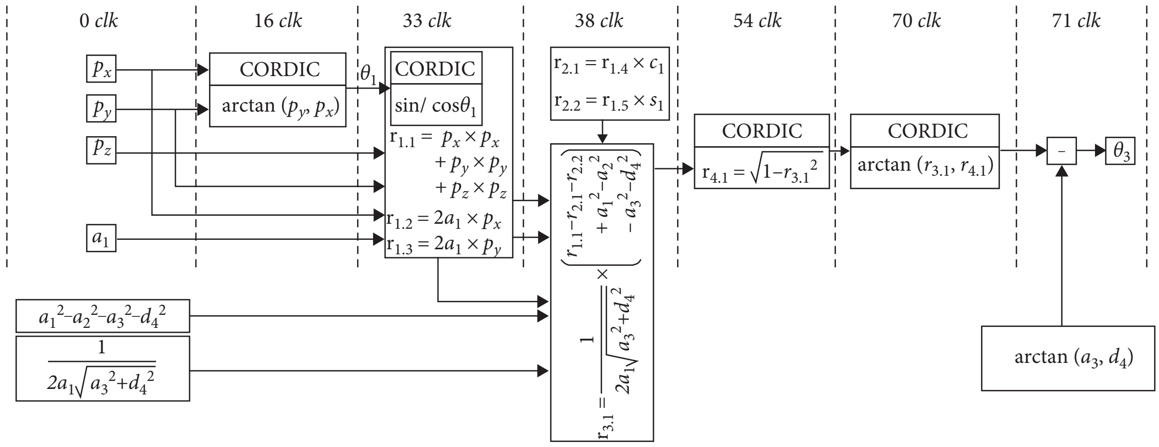


FIGURE 4: Computational model of the forward kinematic solution.



0 clk
16 clk
33 clk
38 clk
54 clk
70 clk
71 clk
p_x
p_y
p_z
a_1
CORDIC
arctan (p_y, p_x)
θ_1
CORDIC
sin/ cosθ_1
$r_{1.1} = p_x \times p_x + p_y \times p_y + p_z \times p_z$
$r_{1.2} = 2a_1 \times p_x$
$r_{1.3} = 2a_1 \times p_y$
$r_{2.1} = r_{1.4} \times c_1$
$r_{2.2} = r_{1.5} \times s_1$
$r_{3.1} = \frac{1}{2a_1\sqrt{a_3^2+d_4^2}} \times \left(r_{1.1} - r_{2.1} - r_{2.2} + a_1^2 - a_2^2 - a_3^2 - d_4^2\right)$
CORDIC
$r_{4.1} = \sqrt{1 - r_{3.1}^2}$
CORDIC
arctan $(r_{3.1}, r_{4.1})$
-
θ_3
$a_1^2 - a_2^2 - a_3^2 - d_4^2$
$\frac{1}{2a_1\sqrt{a_3^2+d_4^2}}$
arctan (a_3, d_4)

FIGURE 5: Calculation model of inverse kinematic solution joint angle.

process is parallelized by register leveling, and the computational models of joint angle θ_1 and θ_3 are shown in Figure 5, where θ_i is the joint angle, c_i means $\cos\theta_i$, s_i means $\sin\theta_i$, $r_{i.j}$ is procedure calculated value. The denominator term of the process data K is converted into a fixed-point multiplication that can be executed by the hard-core multiplier. The results of $1/\left[2a_1(a_3^2 + d_4^2)^{1/2}\right]$, $a_1^2 - a_2^2 - a_3^2 - d_4^2$, and $\arctan(a_3, d_4)$ are obtained by pre-computation, and the resultant values are used directly in the calculation model. The parameters required for the trigonometric function of joint angle θ_1 and the solution of joint angle θ_3 are calculated in parallel to reduce the time consumption of joint angle solution.

Referring to the computational model in Figure 5, the computation of the entire kinematic inverse solution is completed. The parallel pipeline design of the entire kinematic

Table 2: Comparison of calculation time and calculation error.

Test platform	Architecture	Main frequency	Computation time	Computation errors
i5-8500	X86	3 GHz	182 μs	—
nRF52832	ARM	64 MHz	5.28 ms	—
TMS320C6713B	DSP	50 MHz	495 μs	—
EP4CE115	FPGA	50 MHz	3.48 μs	$<10^{-5}$
XC7A200T	FPGA	50 MHz	4.68 μs	$<10^{-5}$

Note: The data of the ARM platform was obtained from [24], the data of the DSP platforms was obtained from [32], the data based on FPGA (EP4CE115) platforms was obtained from [23].

inverse solution is completed by inserting the pipeline register into the computational model, and the pipeline period is 208 *clk*. Therefore, the computation time of the inverse kinematic solution is 208 *clk*.

5. Experimental Verification

To verify the effectiveness of the computational method proposed in this paper, the algorithm is experimented on XC7A200T FPGA platform, where the joint angles θ_i and linkage parameters a_i, α_i, and d_i are represented by 21-bit unsigned numbers, and the poses ${}_6^0T$ are represented by 24-bit signed numbers, which are calculated using $Q20$ fixed points.

5.1. Experiments on Computational Time and Computational Errors. The computational time is obtained by inserting the integrated logic analyzer (ILA) IP into the program, configuring the ILA IP to capture program run signal and result output signal, and then calculating the time difference between the two signals to obtain the time consumed. The computation time includes the computation time of the forward kinematic solution and the inverse kinematic solution. The computational error is based on the results of the X86 architecture platform, where the computation time on X86 architecture platform was deaveraged over 10,000 computations. The computational results of the algorithm in this paper are obtained by inserting the virtual input output (Vio) IP core into the FPGA. The input value of the forward solution computation module is the angle of the robot's joint angle after performing the fixed-point transformation, while the inverse solution computation module inputs the elements of the position matrix into the Vio IP core after performing the fixed-point transformation.

The experimental results obtained in this paper are compared with other methods on X86 architecture platforms, advanced reduced instruction set (RISC) machine (ARM) platforms, DSP platforms, and FPGA platforms as shown in Table 2.

5.2. Discussion. The parallel computation method proposed in this paper reduces the computation time by more than 100 times compared with other software computation schemes for ARM or DSP platforms, and significantly improves the computation efficiency of the kinematic equations. Compared with the study by Celik et al. [19], the computation in this paper takes more time, which is due to the application object of [19] is a 4-degree-of-freedom SCARA robot, the kinematic equations solving steps and computational complexity of SCARA robot are less than that of this paper's 6-degree-of-freedom robot, and importantly this paper proposes the method to be able to carry out the parallel flow computation, the computational efficiency is far more than that in a study by Çelik et al. [19]. The results of converting the fixed-point results of the kinematic forward and inverse solutions to floating-point numbers are compared with those of the software implementation scheme using single-precision floating-point numbers, the computational errors are less than 10^{-5}, and the result is same as in a study by Çelik et al. [19].

It is worth mentioning that the computational time of the computation model proposed in this paper is fixed, which still ensures that the computation period remains unchanged when the data bit width is increased, the computational accuracy will be improved accordingly, but the logic resource consumption of the FPGA will be doubled as well.

6. Conclusions

In this paper, we address the problem of high-real-time computation delay of kinematic equations of multijoint robots, study the hardware logic computation method of kinematic equations based on FPGA, and propose a parallel solution method of kinematic equations based on CORDIC algorithms. In addition, the computational delay of kinematic forward and inverse solutions is reduced to microsecond level, and the solution process is designed as parallel pipeline computation to further improve the computational efficiency.

The parallel solution method proposed in this study is of great importance in the motion control of space robots and ultra-high-speed robots. In future work, the servo control algorithm and the parallel solution method proposed in this paper will be considered for implementation in a single FPGA.

Data Availability

Data were deposited in a public repository.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study is financially supported by the National Natural Science Foundation of China (202014353).

References

[1] K. Merckaert, B. Convens, C.-J. Wu, A. Roncone, M. M. Nicotra, and B. Vanderborght, "Real-time motion control of robotic manipulators for safe human–robot coexistence," *Robotics and Computer-Integrated Manufacturing*, vol. 73, Article ID 102223, 2022.

[2] L. Wang, L. Chen, Z. Shao, L. Guan, and L. Du, "Analysis of flexible supported industrial robot on terminal accuracy," *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, pp. 1–12, 2018.

[3] Y. Kung, B. T. H. Linh, M. Wu, F. Lee, and W. Chen, "FPGA-realization of inverse kinematics control IP for articulated and SCARA robot," in *Design and Computation of Modern Engineering Materials*, vol. 54 of *Advanced Structured Materials*, pp. 205–213, Springer, Cham, 2014.

[4] H. Peng, *Research on 6-DOF tandem robot kinematic algorithm and its control system implementation*, Master Dissertation, Hefei University of Technology, 2016.

[5] J. J. Craig, *Introduction to Robotics*, Machine Industry Press, 2018.

[6] X. Shao, D. Sun, and J. K. Mills, "A new motion control hardware architecture with FPGA-based IC design for robotic manipulators," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, pp. 3520–3525, IEEE, Orlando, FL, USA, 2006.

[7] N. Li, Y. Chen, Y. Shi, F. Gao, J. Che, and J. Chen, "Development of a DSP based control system for a parallel high-quality tea plucking robot," in *American Society of Agricultural and Biological Engineers Annual International Meeting 2015*, pp. 1188–1195, Curran Associates, Inc., 2015.

[8] J.-W. Lee and S. Jung, "A tutorial on control implementation for a collaborative robot in joint space using DSPs," *Journal of Institute of Control, Robotics and Systems*, vol. 28, no. 1, pp. 28–38, 2022.

[9] E. S. Fiestas and S. G. Prado, "Motion control of a cartesian robot using a dual-core ARM cortex-A9 system-on-chip FPGA," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pp. 1–6, IEEE, Curitiba, Brazil, 2017.

[10] Y. Takaki, K. Nagasu, S. Abiko, M. Watanabe, and K. Sano, "FPGA implementation of a robot control algorithm," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1571–1574, IEEE, Zaragoza, Spain, 2019.

[11] Y. Ishida, T. Morie, and H. Tamukoh, "A hardware intelligent processing accelerator for domestic service robots," *Advanced Robotics*, vol. 34, no. 14, pp. 947–957, 2020.

[12] C. V. Duy, "Industrial robot arm controller based on programmable system-on-chip device," *FME Transactions*, vol. 49, no. 4, pp. 1025–1034, 2021.

[13] R. Chand, R. P. Chand, M. Assaf, P. R. Naicker, S. V. Narayan, and A. F. Hussain, "Embedded FPGA-based motion planning and control of a dual-arm car-like robot," in *2022 IEEE 7th Southern Power Electronics Conference (SPEC)*, pp. 1–6, IEEE, Nadi, Fiji, 2022.

[14] H. Gürsoy and M.Ö. Efe, "Control system implementation on an FPGA platform," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 425–430, 2016.

[15] R. Zhang, J. Zhang, Y. Dai, H. Shang, and Y. Li, "Forward and inverse kinematics of the robot based on FPGA," *Journal of Nankai University (Natural Science Edition)*, vol. 51, no. 6, pp. 18–23, 2018.

[16] Y.-S. Kung, M.-K. Wu, H. L. B. Thi, T.-H. Jung, F.-C. Lee, and W.-C. Chen, "FPGA-based hardware implementation of arctangent and arccosine functions for the inverse kinematics of robot manipulator," *Engineering Computations*, vol. 31, no. 8, pp. 1679–1690, 2014.

[17] R. Wei, M. H. Jin, J. J. Xia, Z. W. Xie, and H. Liu, "Reconfigurable parallel VLSI co-processor for space robot using FPGA," in *2006 IEEE International Conference on Robotics and Biomimetics*, pp. 374–379, IEEE, 2006.

[18] Y. Zhang, H. Sun, Q. Jia, and G. Shi, "Kinematics control for a 6-DOF space manipulator based on ARM processor and FPGA co-processor," in *2008 6th IEEE International Conference on Industrial Informatics*, pp. 129–134, IEEE, Daejeon, Korea (South), 2008.

[19] B. Çelik, A. Ak, and V. Topuz, "Field programmable gate arrays based real time robot arm inverse kinematic calculations and visual servoing," *ELECTRICA*, vol. 18, pp. 143–150, 2018.

[20] Manupotisreenivasulu and T. Meenpal, "Efficient MUX based CORDIC on FPGA for signal processing application," in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–6, IEEE, Coimbatore, India, 2019.

[21] G. Evangelista, C. Olaya, and E. Rodriguez, "Fully-pipelined CORDIC-based FPGA realization for a 3-DOF hexapod-leg inverse kinematics calculation," in *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pp. 237–242, IEEE, Beijing, China, 2018.

[22] M. Petko, K. Gac, G. Karpiel, and G. Gora, "Acceleration of parallel robot kinematic calculations in FPGA," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, pp. 34–39, IEEE, Cape Town, South Africa, 2013.

[23] W.-C. Chen, C.-S. Chen, F.-C. Lee, and Y.-S. Kung, "FPGA-realization of the kinematics IP for SCARA robot," *Microsystem Technologies*, vol. 27, no. 4, pp. 1075–1090, 2021.

[24] X. Fan, H. Yan, and D. He, "nverse kinematics of robot based on zynq platform," *Microcontroller and Embedded System Applications*, vol. 17, no. 2, pp. 18–22, 2017.

[25] M. M. Alam, S. Ibaraki, and K. Fukuda, "Kinematic modeling of six-axis industrial robot and its parameter identification: a tutorial," *International Journal of Automation Technology*, vol. 15, no. 5, pp. 599–610, 2021.

[26] M.-A. Cabrera-Rufino, J.-M. Ramos-Arreguín, J. Rodríguez-Reséndiz, E. Gorrostieta-Hurtado, and M.-A. Aceves-Fernandez, "Implementation of ANN-based auto-adjustable for a pneumatic servo system embedded on FPGA," *Micromachines*, vol. 13, no. 6, Article ID 890, 2022.

[27] M.-A. Martinez-Prado, J. Rodriguez-Resendiz, R.-A. Gomez-Loenzo, G. Herrera-Ruiz, and L.-A. Franco-Gasca, "An FPGA-based open architecture industrial robot controller," *IEEE Access*, vol. 6, pp. 13407–13417, 2018.

[28] D. He, F. Liu, and F. Wang, "Optimal design of industrial robot kinematics algorithm," *Journal of Physics: Conference Series*, vol. 1624, no. 4, Article ID 042029, 2020.

[29] D. L. Pieper, *The kinematics of manipulators under computer control*, Phd Thesis, Stanford University, 1968.

[30] Y. Zhong, J. Wu, X. Liu, P. Gao, and X. Duo, "A high-precision inverse tangent solution based on CORDIC algorithm," *Applications of Electronics Technology*, vol. 48, no. 1, pp. 12–17, 2022.

[31] J. S. Walther, “A unified algorithm for elementary functions,” in *AFIPS ’71 (Spring): Proceedings of the May 18-20, 1971, Spring Joint Computer Conference*, pp. 379–385, Association for Computing Machinery, 1971.

[32] Y. Zhang, *Design and implementation of minimally invasive surgical robot control system based on CAN network*, Master Dissertation, Nankai University, 2015.