We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS

BOOK
CITATION
INDEX

INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

FPGA-Realization of a Motion Control IC for Robot Manipulator

Ying-Shieh Kung and Chia-Sheng Chen
Southern Taiwan University
Taiwan

1. Introduction

Robotic control is currently an exciting and highly challenging research focus. Several solutions for implementing the control architecture for robots have been proposed (Kabuka et al., 1988; Yasuda, 2000; Li et al., 2003; Oh et al., 2003). Kabuka et al. (1998) apply two high-performance floating-point signal processors and a set of dedicated motion controllers to build a control system for a six-joint robots arm. Yasuda (2000) adopts a PC-based microcomputer and several PIC microcomputers to construct a distributed motion controller for mobile robots. Li et al. (2003) utilize an FPGA (Field Programmable Gate Array) to implement autonomous fuzzy behavior control on mobile robot. Oh et al. (2003) present a DSP (Digital Signal Processor) and a FPGA to design the overall hardware system in controlling the motion of biped robots. However, these methods can only adopt PC-based microcomputer or the DSP chip to realize the software part or adopt the FPGA chip to implement the hardware part of the robotic control system. They do not provide an overall hardware/software solution by a single chip in implementing the motion control architecture of robot system.

For the progress of VLSI technology, the Field programmable gate arrays (FPGAs) have been widely investigated due to their programmable hard-wired feature, fast time-to-market, shorter design cycle, embedding processor, low power consumption and higher density for implementing digital system. FPGA provides a compromise between the special-purpose ASIC (application specified integrated circuit) hardware and general-purpose processors (Wei et al., 2005). Hence, many practical applications in motor control (Zhou et al., 2004; Yang et al., 2006; Monmasson & Cirstea, 2007) and multi-axis motion control (Shao & Sun, 2005) have been studied, using the FPGA to realize the hardware component of the overall system.

The novel FPGA (Field Programmable Gate Array) technology is able to combine an embedded processor IP (Intellectual Property) and an application IP to be an SoPC (System-on-a-Programmable-Chip) developing environment (Xu et al., 2003; Altera, 2004; Hall & Hamblen, 2004), allowing a user to design an SoPC module by mixing hardware and software. The circuits required with fast processing but simple computation are suitable to be implemented by hardware in FPGA, and the highly complicated control algorithm with heavy computation can be realized by software in FPGA. Results that the software/hardware co-design function increase the programmable, flexibility of the designed digital system and reduce the development time. Additionally, software/hardware parallel processing enhances the controller performance. Our previous works (Kung & Shu, 2005; Kung et al., 2006; Kung &

Tsai, 2007) have successfully applied the novel FPGA technology to the servo system of PMSM drive, robot manipulator and X-Y table.

To exploit the advantages, this study presents a fully digital motion control IC for a five-axis robot manipulator based on the novel FPGA technology, as in Fig. 1(a). Firstly, the mathematical model and the servo controller of the robot manipulator are described. Secondly, the inverse kinematics and motion trajectory is formulated. Thirdly, the circuit being designed to implement the function of motion control IC is introduced. The proposed motion control IC has two IPs. One IP performs the functions of the motion trajectory for robot manipulator. The other IP performs the functions of inverse kinematics and five axes position controllers for robot manipulator. The former is implemented by software using Nios II embedded processor due to the complicated processing but low sampling frequency control (motion trajectory: 100Hz). The latter is realized by hardware in FPGA owing to the requirements of high sampling frequency control (position loop:762Hz, speed loop:1525Hz, PWM circuit: 4~8MHz) but simple processing. To reduce the usage of the logic gates in FPGA, an FSM (Finite State Machine) method is presented to model the overall computation of the inverse kinematics, five axes position controllers and five axes speed controllers. Then the VHDL (VHSIC Hardware Description Language) is adopted to describe the circuit of FSM. Hence, all functionality required to build a fully digital motion controller for a five-axis robot manipulator can be integrated in one FPGA chip. As the result, hardware/software co-design technology can make the motion controller of robot manipulator more compact, flexible, better performance and less cost. The FPGA chip employed herein is an Altera Stratix II EP2S60F672C5ES (Altera, 2008), which has 48,352 ALUTs (Adaptive Look-Up Tables), maximum 718 user I/O pins, total 2,544,192 RAM bits. And a Nios II embedded processor which has a 32-bit configurable CPU core, 16 M byte Flash memory, 1 M byte SRAM and 16 M byte SDRAM is used to construct the SoPC development environment. Finally, an experimental system included by an FPGA experimental board, five sets of inverter and a Movemaster RV-M1 micro robot produced by Mitsubishi is set up to verify the correctness and effectiveness of the proposed FPGA-based motion control IC.

2. Important

The typical architecture of the conventional motion control system for robot manipulator is shown in Fig. 1(b), which consists of a central controller, five sets of servo drivers and one robot manipulator. The central controller, which usually adopts a float-pointed processor, performs the function of motion trajectory, inverse kinematics, data communication with servo drivers and the external device. Each servo driver uses a fixed-pointed processor, some specific ICs and an inverter to perform the functions of position control at each axis of robot manipulator and the data communication with the central controller. Data communication media between them may be an analog signal, a bus signal or a serial asynchronous signal. Therefore, once the central controller receives a motion command from external device. It will execute the computation of motion trajectory and inverse kinematics, generate five position commands and send those signals to five servo drivers. Each servo driver received the position command from central controller will execute the position controller to control the servo motor of manipulator. As the result, the motion trajectory of robot manipulator will follow the prescribed motion command. However, the motion control system in Fig .1 (b) has some drawbacks, such as large volume, easy effected by the noise, expensive cost, inflexible, etc. In addition, data communication and handshake protocol between the central controller and servo drivers slow down the system executing speed. To improve aforementioned drawbacks,

based on the novel FPGA technology, the central controller and the controller part of five servo drivers in Fig. 1 (b) are integrated into a motion control IC in this study, which is shown in Fig. 1(a). The proposed motion control IC has two IPs. One IP performs the functions of the motion trajectory by software. The other IP performs the functions of inverse kinematics and five axes position controllers by hardware. Hence, hardware/software co-design technology can make the motion controller of robot manipulator more compact, flexible, better performance and less cost. The detailed design methodology is described in the following sections, and the contributions of this study will be illustrated in the conclusion section.
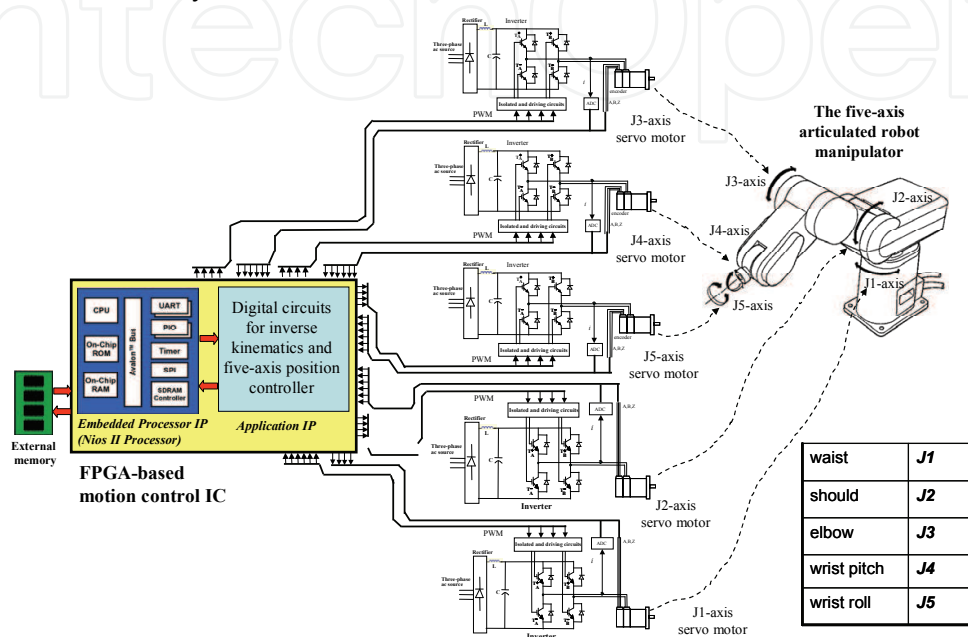
(a)
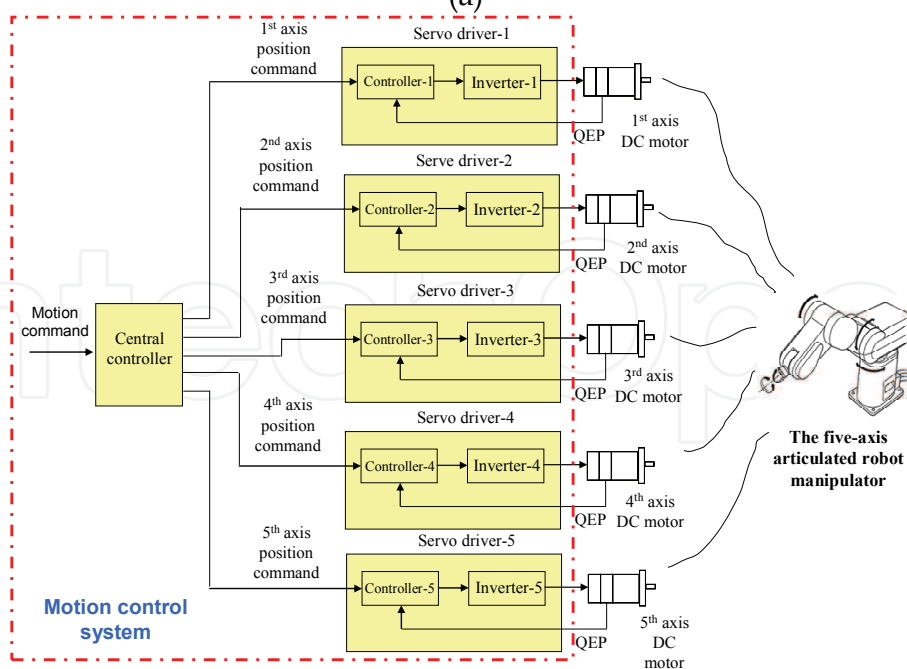
(b)

Figure 1. (a) The proposed architecture of an FPGA-based motion control system for robot manipulator (b) the conventional motion control system for robot manipulator

3. System description and design method of robot manipulator

3.1 Mathematical model of a robot manipulator with actuator

The dynamic equation of the n-link manipulator is given by (Lewis et al., 1993):

$$M(\theta)\ddot{\theta} + V_m(\theta,\dot{\theta})\dot{\theta} + F(\dot{\theta}) + G(\theta) = \tau \tag{1}$$

where $M(\theta)$ denotes the inertial matrix; $V_m(\theta,\dot{\theta})$ denotes the Coriolis/centripetal vector; $G(\theta)$ denotes the gravity vector; and $F(\dot{\theta})$ denotes the friction vector. The terms θ, $\dot{\theta}$, $\ddot{\theta}$ and τ represent the n-vector of the position, the velocity, the acceleration and the generalized forces, respectively; and are set to R^n. The dynamic model of the ith axis DC motor drive system for the robot manipulator is expressed as

$$L_{a_i}\frac{di_i}{dt} + R_{a_i}i_i = v_i - K_{b_i}\dot{\theta}_{M_i} \tag{2}$$

$$\tau_{M_i} = K_{M_i}i_i \tag{3}$$

$$J_{M_i}\ddot{\theta}_{M_i} + B_{M_i}\dot{\theta}_{M_i} = \tau_{M_i} - \tau_{L_i} \tag{4}$$

where L_{a_i}, R_{a_i}, i_i, v_i, K_{b_i}, K_{M_i} denote inductance, resistance, current, voltage, voltage constant and current constant, respectively. Terms J_{M_i}, B_{M_i}, τ_{M_i} and τ_{L_i} denote the inertial, viscous and generated torque of the motor, and the load torque to the ith axis of the robot manipulator, respectively. Consider a friction F_{M_i} added to each arm link and replace the load torque τ_{L_i} by an external load torque through a gear ratio r_i, giving $\tau_{L_i} = r_i\tau_i$ and assume that the inductance term in (2) is small and can be ignored. Then, (2) to (4) can be arranged and simplified by the following equation.

$$J_{M_i}\ddot{\theta}_{M_i} + (B_{M_i} + \frac{K_{M_i}K_{b_i}}{R_{a_i}})\dot{\theta}_{M_i} + F_{M_i} + r_i\tau_i = \frac{K_{M_i}}{R_{a_i}}v_i \tag{5}$$

Therefore, the dynamics of the dc motors that drive the arm links are given by the n decoupled equations

$$J_M\ddot{\theta}_M + B\dot{\theta}_M + F_M + R\tau = K_M v \tag{6}$$

with

$$\begin{aligned}
&\theta_M = vec\{\theta_{M_i}\}, \; J_M = diag\{J_{M_i}\} \\
&B = diag\{B_{M_i} + K_{b_i}K_{M_i}/R_{a_i}\} \triangleq diag\{B_i\} \\
&R = diag\{r_i\}, \;\; K_M = diag\{K_{M_i}/R_{a_i}\} \\
&v = vce(v_i), \;\; \tau = vec(\tau_i), \;\; F_M = vec(F_{M_i})
\end{aligned} \tag{7}$$

The dynamic equations including the robot manipulator and DC motor can be obtained by combining (1) and (6). The gear ratio of the coupling from the motor i to arm link i is given by r_i, which is defined as

$$\theta_i = r_i\theta_{M_i} \text{ or } \theta = R\theta_M \tag{8}$$

Hence, substituting (8) into (6), then into (1), it can be obtained by

$$(J_M + R^2 M(\theta))\ddot{\theta} + (B + R^2 V_m(\theta,\dot{\theta}))\dot{\theta} + R F_M(\dot{\theta}) + R^2 F(\dot{\theta}) + R^2 G(\theta) = R K_M v \qquad (9)$$

The gear ratio r_i of a commercial robot manipulator is usually small ($< 1/100$) to increase the torque value. Therefore, the formula can be simplified. From (9), the dynamic equation combining the motor and arm link is expressed as

$$(J_{M_i} + r_i^2 m_{ii})\ddot{\theta}_i + B_i \dot{\theta}_i + r_i F_{M_i} = \frac{r_i K_{M_i}}{R_{a_i}} v_i - r_i^2 d_i \qquad (10)$$

where

$$d_i = \sum_{j \neq i} m_{ij} \ddot{\theta}_j + \sum_{j,k} V_{jki} \dot{\theta}_j \dot{\theta}_k + F_i + G_i \qquad (11)$$
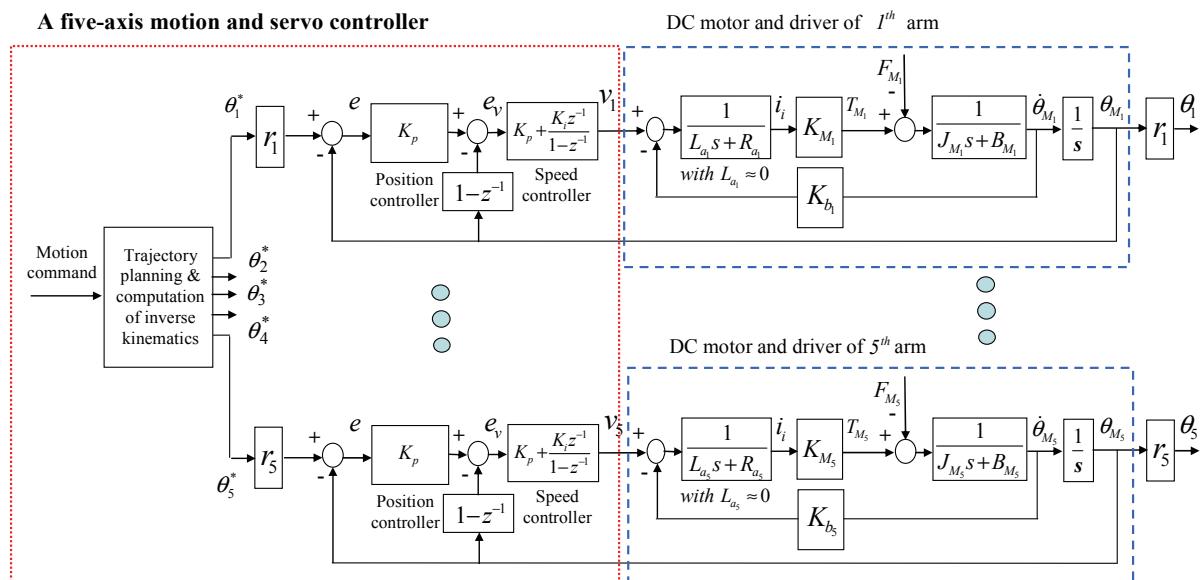

Figure 2 The block diagram of motion and servo controller with DC motor for a five-axis robot manipulator

Since the gear ratio r_i is small in a commercial robot manipulator, the r_i^2 term in (10) can be ignored, and (10) can be simplified as

$$J_{M_i}\ddot{\theta}_i + B_i \dot{\theta}_i + r_i F_{M_i} = \frac{r_i K_{M_i}}{R_{a_i}} v_i \qquad (12)$$

Substituting (7) and (8) into (12), yields

$$J_{M_i}\ddot{\theta}_{M_i} + (B_{M_i} + \frac{K_{M_i} K_{b_i}}{R_{a_i}})\dot{\theta}_{M_i} + F_{M_i} = \frac{K_{M_i}}{R_{a_i}} v_i \qquad (13)$$

If the gear ratio is small, then the control block diagram combining arm link, motor actuator, position loop P controller, speed loop PI controller, inverse kinematics and trajectory planning for the servo and motion controller in a five-axis robot manipulator can be represented in Fig. 2.

In Fig.2, the digital PI controllers in the speed loop are formulated as follows.

$$v_p(n) = k_p \, e(n) \tag{14}$$

$$v_i(n) = v_i(n-1) + k_i \, e(n-1) \tag{15}$$

$$v(n) = v_p(n) + v_i(n) \tag{16}$$

Where the k_p, k_i are P-controller gain and I-controller gain, the $e(n)$ is the error between command and sensor value and the $v(n)$ is the PI controller output.

3.2 Computational of the inverse kinematics for robot manipulator

Figure 3 shows the link coordinate system of the five-axis articulated robot manipulator using the Denavit-Hartenberg convention. Table 1 illustrates the values of the kinematics parameters. The inverse kinematics of the articulated robot manipulator will transform the coordinates of robot manipulator from Cartesian space R^3 (x,y,z) to the joint space R^5 ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$) and it is described in detail by author of (Schilling, 1998). The computational procedure is as follows.

Step1: $$\theta_1 = \theta_5 = a \tan 2(y, x) \tag{17}$$

Step2: $$b = \sqrt{x^2 + y^2} \tag{18}$$

Step3: $$\theta_3 = \cos^{-1}\left(\frac{b^2 + (d_1 - d_5 - z)^2 - a_2^2 - a_3^2}{2a_2 a_3} \right) \tag{19}$$

Step4: $$S_2 = (a_2 + a_3 \cos\theta_3)(d_1 - d_5 - z) - a_3 b \sin\theta_3 \tag{20}$$

Step5: $$C_2 = (a_2 + a_3 \cos\theta_3)b + a_3 \sin\theta_3 (d_1 - d_5 - z) \tag{21}$$

Step6: $$\theta_2 = a \tan 2(S_2, C_2) \tag{22}$$

Step7: $$\theta_4 = -\theta_2 - \theta_3 \tag{23}$$

where a_1, a_5, d_1, d_5 are Denavit-Hartenberg parameters and *atan2* function refers to Table 2.
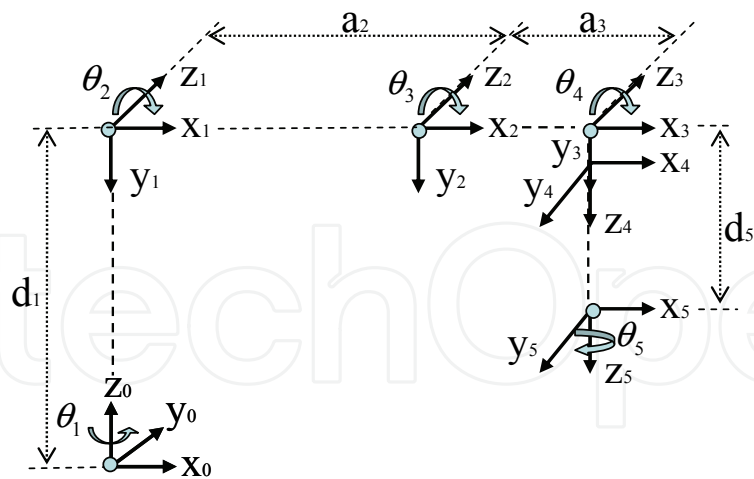
Figure 3. Link coordinates of a five-axis robot manipulator

	θ	d	a	α
1	θ_1	d_1 (300 mm)	(0 mm)	$\alpha_1(-\pi/2)$
2	θ_2	(0 mm)	a_2 (250 mm)	0
3	θ_3	(0 mm)	a_3 (160 mm)	0
4	θ_4	(0 mm)	(0 mm)	$\alpha_4(-\pi/2)$
5	θ_5	d_5 (72 mm)	(0 mm)	0

Table 1. Denavit-Hartenberg parameters of the robot manipulator

Case	Quadrants	$atan2\ (y, x)$
$x > 0$	1, 4	$arctan\ (y\ /\ x)$
$x = 0$	1, 4	$[sgn\ (y)]\pi/2$
$x < 0$	2, 3	$arctan\ (y\ /\ x) + [sgn\ (y)]\pi$

Table 2. Four-quadrant *arctan* function (Schilling, 1998)

3.3 Planning of the motion trajectory

3.3.1The computation of the point-to-point motion trajectory

To consider the smooth motion when manipulating the robot, the point-to-point motion scheme with constant acceleration/deceleration trapezoid velocity profile is applied in Fig.2. In this scheme, the designed parameters at each axis of robot are the overall displacement $\Delta\theta_i^*$ (degree), the maximum velocity W_i (degree/s), the acceleration or deceleration period T_{acc} and the position loop sampling interval t_d. Therefore, the instantaneous position command q at each sampling interval, based on the trapezoid velocity profile, can be computed as follows:

Step 1: Compute the overall running time. First, compute the maximum running time without considering the acceleration and deceleration:

$$T_1 = Max(\Delta\theta_1^* / W_1, \Delta\theta_2^* / W_2, \Delta\theta_3^* / W_3, \Delta\theta_4^* / W_4, \Delta\theta_5^* / W_5) \tag{24}$$

This T_1 must exceed acceleration time T_{acc}. The acceleration and deceleration design is then considered, and the overall running time is

$$T = Max\ (T_1, T_{acc}) + T_{acc} \tag{25}$$

Step 2: Adjust the overall running time to meet the condition of a multiple of the sampling interval.

$$N' = [T_{acc}/t_d]\ \text{and}\ N = [T/t_d] \tag{26}$$

Where N represents the interpolation number and [.] refers to the Gauss function with integer output value. Hence,

$$T_{acc}' = N' * t_d\ \text{and}\ T' = N * t_d \tag{27}$$

Step 3: Modify the maximum velocity

$$L \underline{\underline{\Delta}} [\Delta\theta_1^*, \Delta\theta_2^*, \Delta\theta_3^*, \Delta\theta_4^*, \Delta\theta_5^*]^T \tag{28}$$

$$W' \underline{\underline{\Delta}} [W_1', W_2', W_3', W_4', W_5']^T = L / (T' - T_{acc}') \tag{29}$$

Step 4 : Calculate the acceleration/deceleration value

$$A = W' / T_{acc}' \tag{30}$$

Step 5 : Calculate the position command, $q = [q_1, q_2, q_3, q_4, q_5]$, at the mid-position

a.   Acceleration region:

$$q = q_{r0} + \frac{1}{2} * A * t^2 \tag{31}$$

where $t = n * t_d$, $0 < n \le N'$, and q_{r0} denotes the initial position.

b.   Constant speed region:

$$q = q_{r1} + W' * t \tag{32}$$

where $t = n * t_d$, $0 < n \le N1$, q_{r1} denotes the final position in the acceleration region and $N1 = N - 2 * N'$.

c.   Deceleration region:

$$q = q_{r2} + (W' * t - \frac{1}{2} * A * t^2) \tag{33}$$

where $t = n * t_d$, $0 < n \le N'$ and q_{r2} denotes the final position in the constant speed region.

Therefore, applying this point-to-point motion control scheme, all joins of robot will rotate with simultaneous starting and stopping.

3.3.2 The computation of the linear and circular motion trajectory
Typical motion trajectories have linear and circular trajectory.
a.   Linear motion trajectory
    The formulation of linear motion trajectory is as follows:

$$x_i = x_{i-1} + \Delta s \, \sin\gamma \cos\alpha \tag{34}$$

$$y_i = y_{i-1} + \Delta s \, \sin\gamma \cos\beta \tag{35}$$

$$z_i = z_{i-1} + \Delta s \, \cos\gamma \tag{36}$$

where Δs, γ, α, β, x_i, y_i, z_i are path increment, angle between z-axis and path vector, angle between x-axis and the vector that path vector projects to x-y plane, angle between y-axis and the vector that path vector projects to x-y plane, x-axis, y-axis and z-axis trajectory command, respectively.   The running speed of the linear motion is determined by Δs.

b.   Circular motion trajectory with constant value in Z-axis
    The formulation of circular motion trajectory with constant value in Z-axis is as follows:

$$\varphi_i = \varphi_{i-1} + \Delta\varphi \tag{37}$$

$$x_i = R\sin(\varphi_i) \tag{38}$$

$$y_i = R\cos(\varphi_i) \tag{39}$$

$$z_i = z_{i-1} \tag{40}$$

where φ_i, $\Delta\varphi$, R, x_i, y_i and z_i are angle, angle increment, radius, x-axis, y-axis and z-axis trajectory command, respectively.   The running speed of the circle motion is determined by $\Delta\varphi$.

4. The proposed FPGA-based motion IC for robot manipulator

Figure 4 illustrates the internal architecture of the proposed FPGA-based motion control IC for a five-axis robot manipulator. The motion control IC, which comprises a Nios II embedded processor IP and a motion control IP, is designed based on the SoPC technology, which is developed by Altera Cooperation. The FPGA chip employed herein is an Altera Stratix II EP2S60F672C5ES, which has 48,352 ALUTs, maximum 718 user I/O pins, total 2,544,192 RAM bits. And a Nios II embedded processor has a 32-bit configurable CPU core,

16 M byte Flash memory, 1 M byte SRAM and 16 M byte SDRAM. A custom software development kit (SDK) consists of a compiled library of software routines for the SoPC design, a Make-file for rebuilding the library, and C header files containing structures for each peripheral.

The Nios II embedded processor depicted in Fig. 4 performs the function of motion command, computation of point-to-point motion trajectory as well as linear and circular motion trajectory in software. The clock frequency of the Nios II processor of is 50MHz. Figure 5 illustrates the flow charts of the main program, subroutine of point-to-point trajectory planning and the interrupt service routine (ISR), where the interrupt interval is designed with 10ms. However, point-to-point motion and trajectory tracking motion are run by different branch program. Because the inverse kinematics module is realized by hardware in FPGA, the computation in Fig. 5 needed to compute the inverse kinematics has to firstly send the x, y, z value from I/O pin of Nios II processor to the inverse kinematics module, wait 2μs then read back the $\theta_1^* \sim \theta_5^*$. All of the controller programs in Fig. 5 are coded in the C programming language; then, through the complier and linker operation in the Nios II IDE (Integrated Development Environment), the execution code is produced and can be downloaded to the external Flash or SDRAM via JTAG interface. Finally, this execution code can be read by Nios II processor IP via bus interface in Fig. 4. Using the C language to develop the control algorithm not only has the portable merit but also is easier to transfer the mature code, which has been well-developed in other devices such as DSP device or PC-based system, to the Nios II processor.
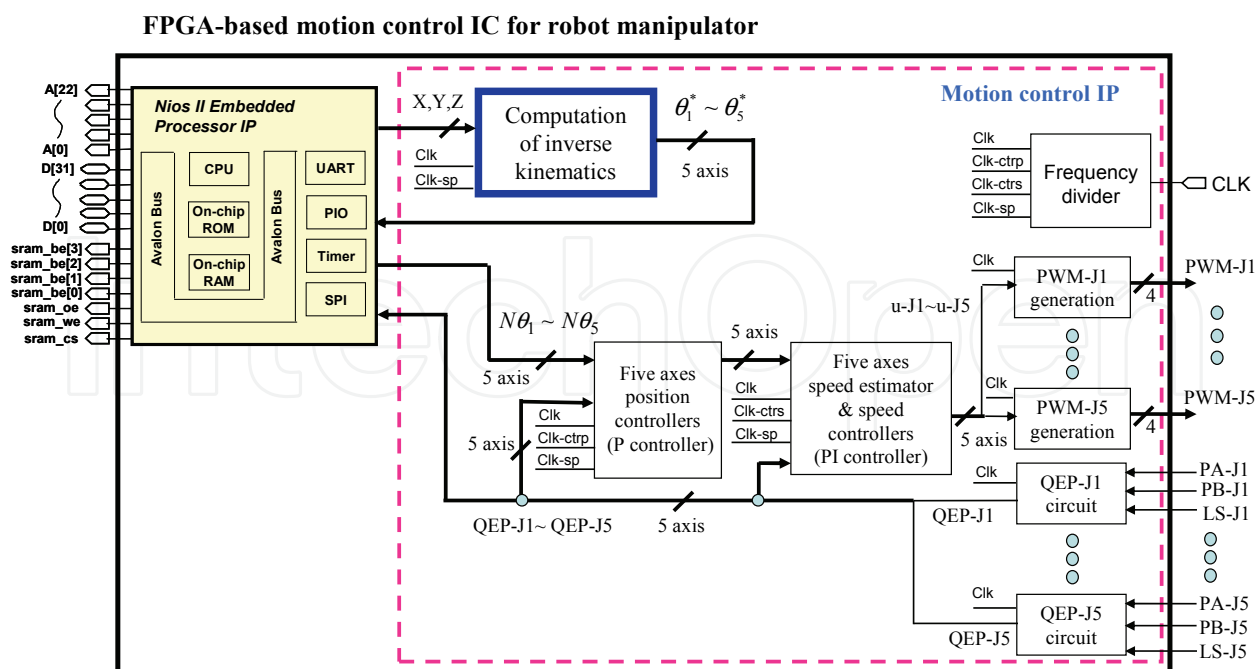

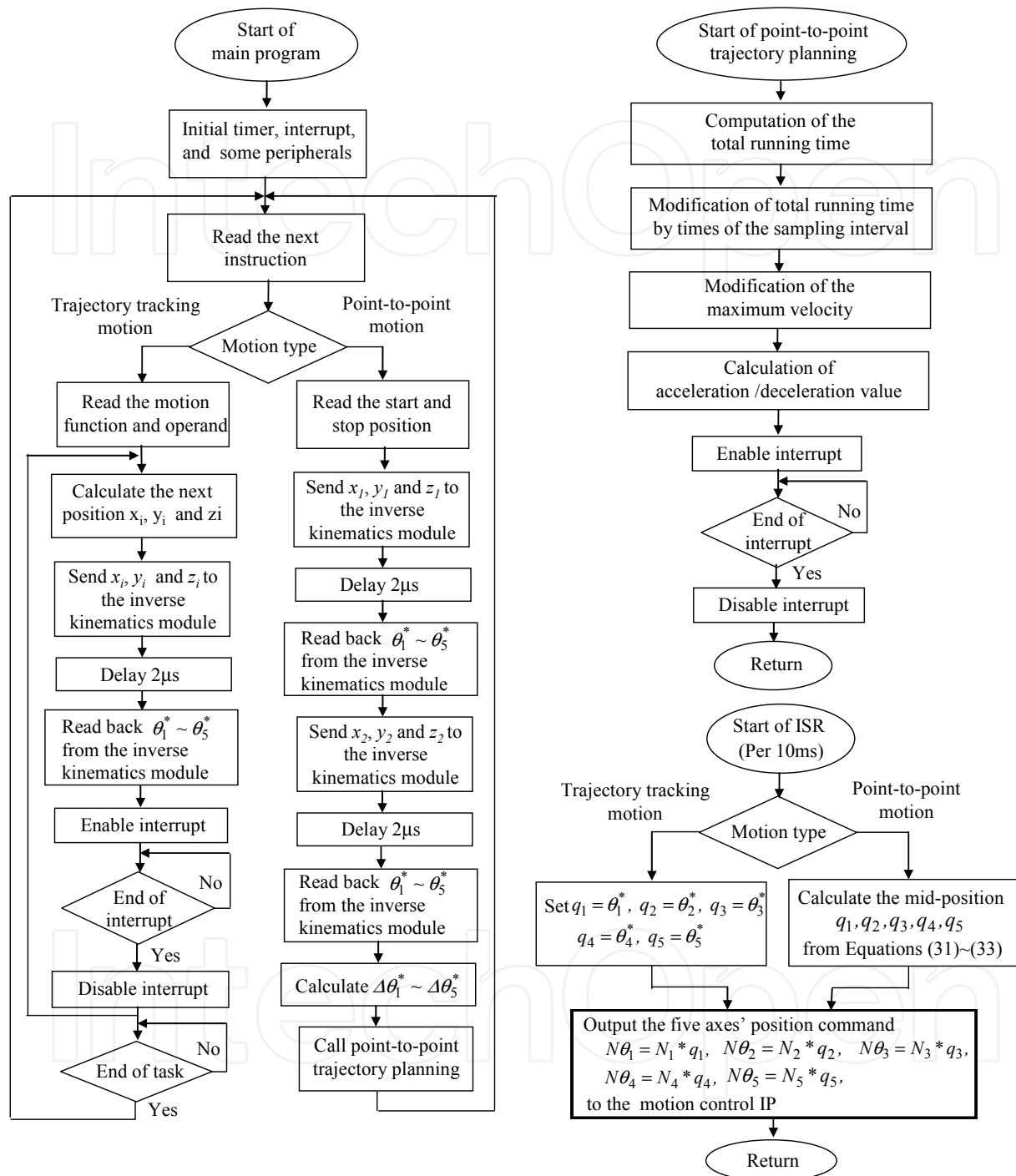Figure 4. Internal architecture of the proposed FPGA-based motion control IC for robot manipulator

Figure 5. Flow charts of main and ISR program in Nios II embedded processor

The motion control IP implemented by hardware, as depicted in Fig. 4, is composed of a frequency divider, an inverse kinematics module, a five-axis position controller module, a five-axis speed estimated and speed controller module, five sets of QEP (Quadrature

Encoder Pulse) module and five sets of PWM (Pulse Width Modulation) module. The frequency divider generates 50MHz (*clk*), 762Hz (*clk_ctrp*), 1525Hz (*clk_ctrs*) and 50MHz (*clk_sp*) to supply all module circuits of the motion control IP in Fig.4. To reduce the usage in FPGA, the FSM method is proposed to design the circuit in the position/speed controller module and the inverse kinematics module, as well as the VHDL is used to describe the behaviors. Figure 6(a) and 6(b) respectively describe the behavior of the single axis *P* controller in the position controller module and the single axis *PI* controller in the speed controller module. There are 4 steps and 12 steps to complete the computation of the single axis *P* controller and the single axis *PI* controller, respectively. And only one multiplier, one adder and one comparator circuits are used in Fig.6. The data type is 16-bit length with Q15 format and 2′s complement operation. To prevent numerical overflow and alleviate windup phenomenon, the output values of I controller and *PI* controller are both limited within a specific range. The step-by-step computation of single axis controller is extended to five axes controller and illustrated in Fig.7. We can find that, it needs 20 steps to complete the computation of overall five axes P controllers in position control in Fig. 7(a), and 60 steps to complete the computation of overall five axes PI controllers in speed control in Fig. 7(b). However, in Fig.7(a) and Fig.7(b), it still need only one multiplier, one adder and one comparator circuits to complete the overall circuit, respectively. Furthermore, because the execution time of per step machine is designed with 20ns (*clk_sp* : 50MHz clock), the total execution time of five axes controller needs 0.4μs and 1.2 μs in position and speed loop control, respectively. And, the sampling frequency in position and speed loop control are respectively designed with 762 Hz (1.31ms) and 1525 Hz (0.655ms) in our proposed servo system of robot manipulator. It apparently reveals that it is enough time to complete the five axes controller during the control interval. Designed with FSM method, although five axes controllers need much computation time than the single axis controller, but the resource usage in FPGA is the same. The circuit of the *QEP* module is shown in Fig.8(a), which consists of two digital filters, a decoder and an up-down counter. The filter is used for reducing the noise effect of the input signals *PA* and *PB*. The pulse count signal *PLS* and the rotating direction signal *DIR* are obtained using the filtered signals through the decoder circuit. The *PLS* signal is a four times frequency pulses of the input signals *PA* or *PB*. The *Qep* value can be obtained using *PLS* and *DIR* signals through a directional up-down counter. Figure 8(b) and (c) are the simulation results for *QEP* module. Figure 8(b) shows the count-up mode where *PA* signal leads to *PB* signal, and Fig. 8(c) shows the count-down mode while *PA* signal lags to *PB* phase signal. The circuit of the *PWM* module is shown in Fig.9(a), which consists of two up-down counters, two comparators, a frequency divider, a dead-band generating circuit, four *AND* logic gates and a *NOR* logic gate. The first up-down counter generates an 18 kHz frequency symmetrical triangle wave. The counter value will continue comparing with the input signal *u* and generating a *PWM* signal through the comparator circuit. To prevent the short circuit occurred while both gates of the inverter are triggered-on at the same time; a circuit with 1.28μs dead-band value is designed. Finally, after the output signals of *PWM* comparator and dead-band comparator through *AND* and *NOR* logic gates circuit, four *PWM* signals with 18 *kHz* frequency and 1.28μs dead-band values are generated and sent out to trigger the IGBT device. Figure 9(b) shows the *PWM* simulated waveforms, and this result demonstrates the correctness of the *PWM* module.
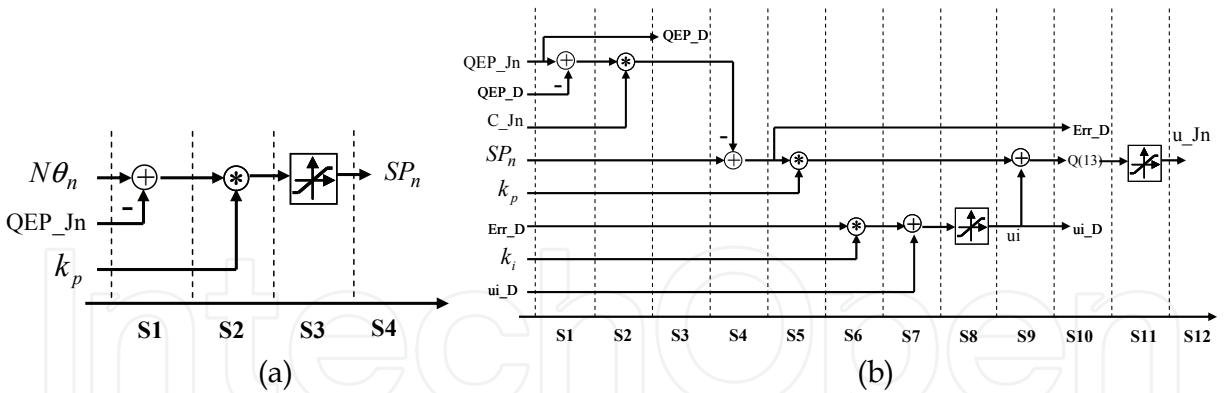
Figure 6 (a) The single axis P controller circuit in the position controller module (b) Speed estimation and single axis PI controller circuit in the speed controller module
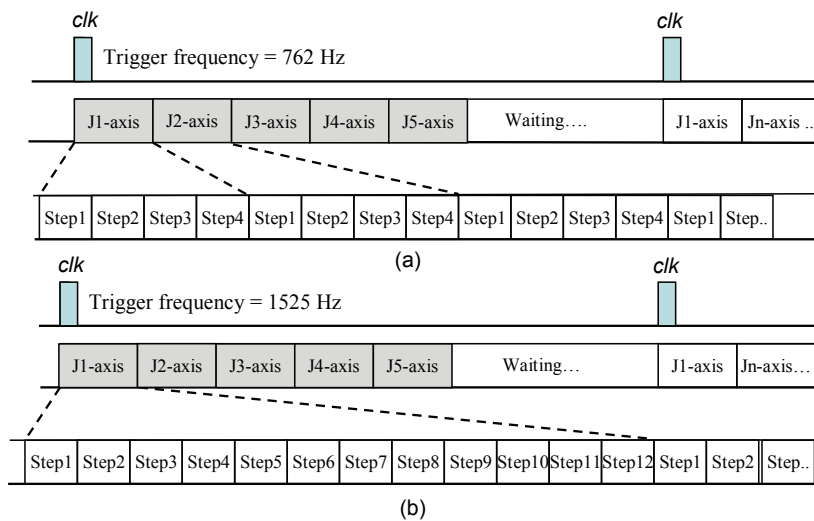

Figure 7. Sequence control signal for five axes (a) position controller (b) speed controller
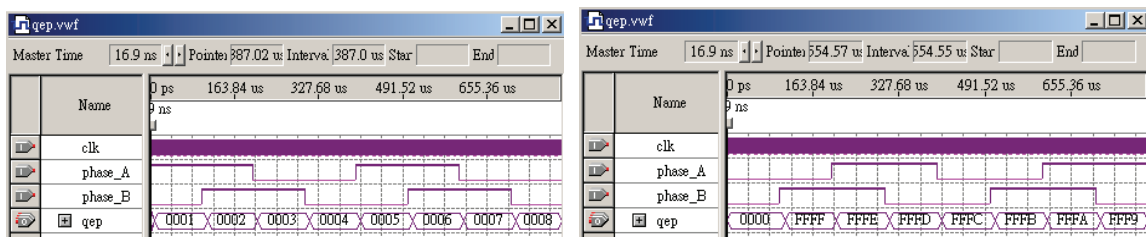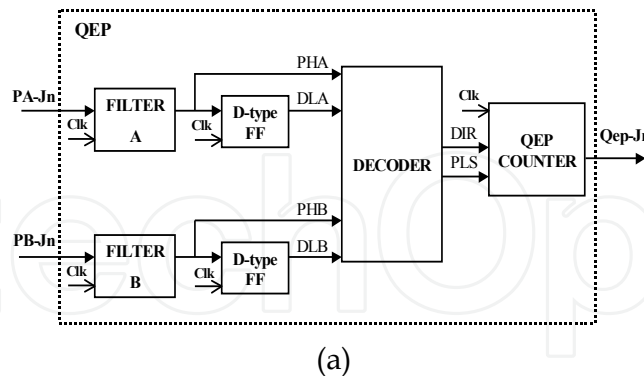

Figure 8. (a) QEP module circuit and its (b) count-up (c) count-down simulation result
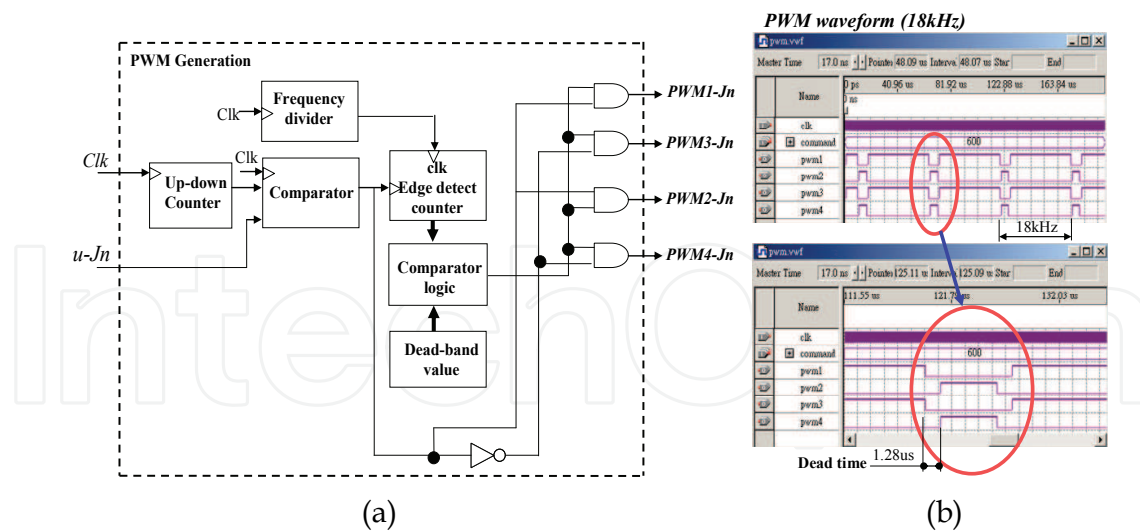
(a)   (b)

Figure 9. (a) PWM module circuit and its (b) simulation result

From the equations of inverse kinematics in (17)~(23), it is hard to understand the designed code of VHDL. Therefore, we reformulate it as follows.

$$v_1 = d_1 - d_5 - z \tag{41}$$

$$\theta_1 = a\tan 2(y, x) \tag{42}$$

$$v_2 = b = \sqrt{x^2 + y^2} \tag{43}$$

$$\theta_5 = \theta_1 \tag{44}$$

$$v_3 = (v_2^2 + v_1^2 - a_2^2 - a_3^2) / 2a_2 a_3 \tag{45}$$

$$\theta_3 = \cos^{-1}(v_3) \tag{46}$$

$$v_4 = sin\theta_3 \tag{47}$$

$$v_5 = a_3 cos\theta_3 + a_2 \tag{48}$$

$$v_6 = a_3 cos\theta_3 \tag{49}$$

$$v_7 = a_3 sin\theta_3 \tag{50}$$

$$S_2 = (a_2 + a_3 \cos\theta_3)(d_1 - d_5 - z) - a_3 b \sin\theta_3 \\ = v_5 \cdot v_1 - v_7 \cdot v_2 \tag{51}$$

$$C_2 = (a_2 + a_3 cos\theta_3)b + a_3 sin\theta_3(d_1 - d_5 - z) \\ = v_5 \cdot v_2 + v_7 \cdot v_1 \tag{52}$$

$$\theta_2 = a\tan 2(S_2, C_2) \tag{53}$$

$$\theta_4 = -\theta_2 - \theta_3 \tag{54}$$

where $v_1 \sim v_7, S_2, C_2$ are temporary variables. The computation of inverse kinematics in (41)~(54) by parallel operation method is resource consumption in FPGA; therefore, an FSM is adopted to model the inverse kinematics and illustrated in Fig. 10, then the VHDL is adopted to describe the circuit of FSM. The data type is 16-bit length with Q15 format and 2's

complement operation. There are total 42 steps with 20ns/step to perform the overall computation of inverse kinematics. The circuits in Fig.10 need two multipliers, one divider, two adders, one component for square root function, one component for arctan function, one component for arcos function, one look-up-table for sin function and some comparators for atan2 function. The multiplier, adder, divider and square root circuit are Altera LPM (Library Parameterized Modules) standard component but the arcos and arctan function are our developed component. The divider and square root circuits respectively need 5 steps executing time in Fig. 10; the arcos and arctan component need 9 steps executing time; others circuit, such as adder, multiplier, LUT etc., needs only one step executing time. Furthermore, to realize the overall computation of the inverse kinematics needs 840ns (20ns/step* 42 steps) computation time. In our previous experience, the computation for inverse kinematics in Nios II processor using C language by software will take 5.6ms. Therefore, the computation of the inverse kinematics realized by hardware in FPGA is 6,666 times faster than those in software by Nios II processor. Finally, to test the correctness of the computation of inverse kinematics in Fig.10, three cases at Cartesian space (0,0,0), (200,100,300) and (300,300,300) are simulated and transformed to the joint space. The simulation results are shown in Fig. 11 that *com_x*, *com_y*, *com_z* denote input commands at Cartesian space, and those *com_1* to *com_5* denote output position commands at joint space. The simulation result in Figure 11 demonstrates the correctness of the proposed design method of the inverse kinematics.
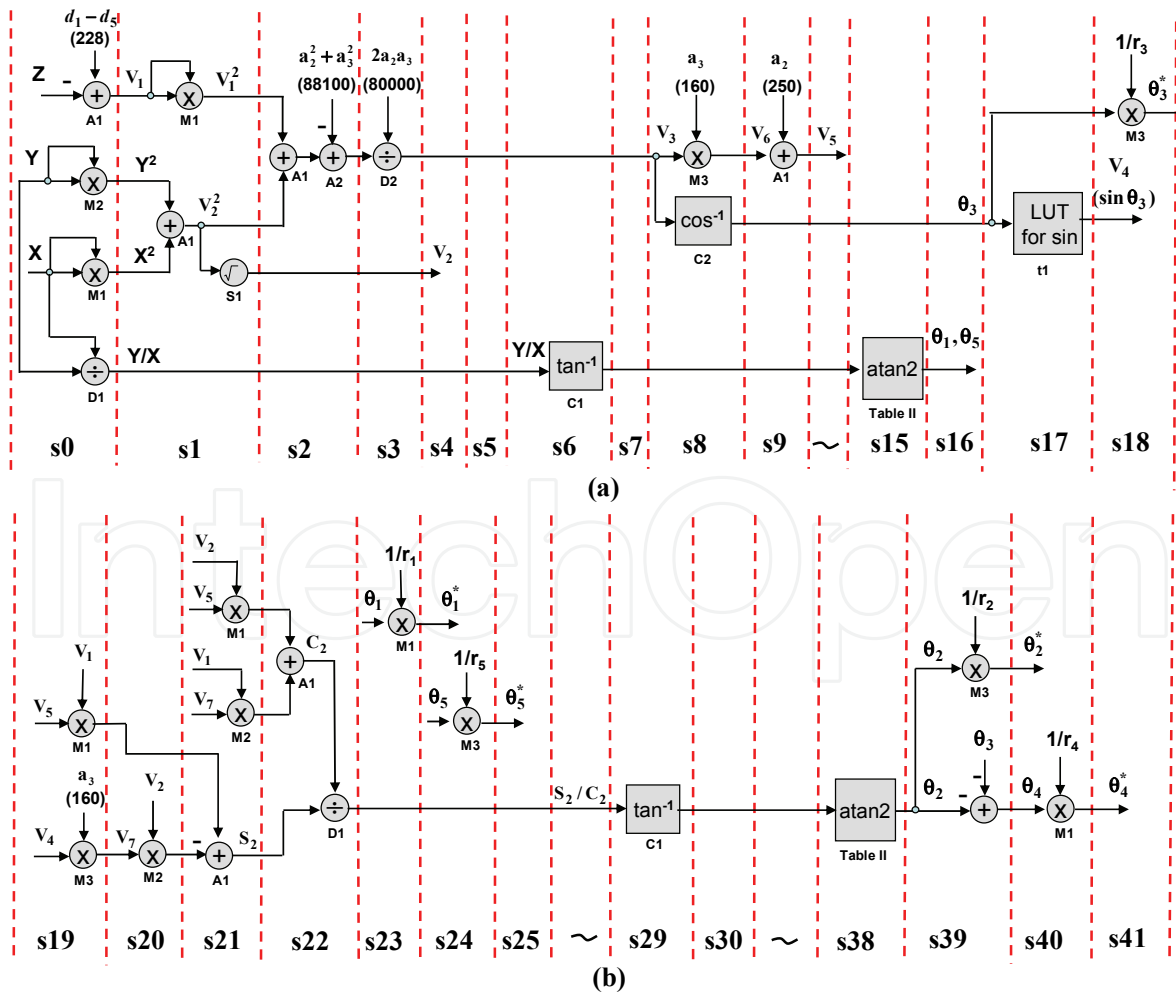

Figure 10. Circuit of computing the inverse kinematics using finite state machine

Finally, the FPGA utility of the motion control IC for robot manipulator in Fig. 4 is evaluated and the result is listed in Table 3. The overall circuits included a Nios II embedded processor (5.1%, 2,468 ALUTs) and a motion control IP (14.4%, 6,948 ALUTs) in Fig. 4, use 19.5% (9,416 ALUTs) utility of the Stratix II EP2S60. Nevertheless, for the cost consideration, the less expensive device - Stratix II EP2S15 (12,480 ALUTs and 419,329 RAM bits) is a better choice. In Fig. 4, the software/hardware program in parallel processing enhances the controller performance of the motion system for the robot manipulator.
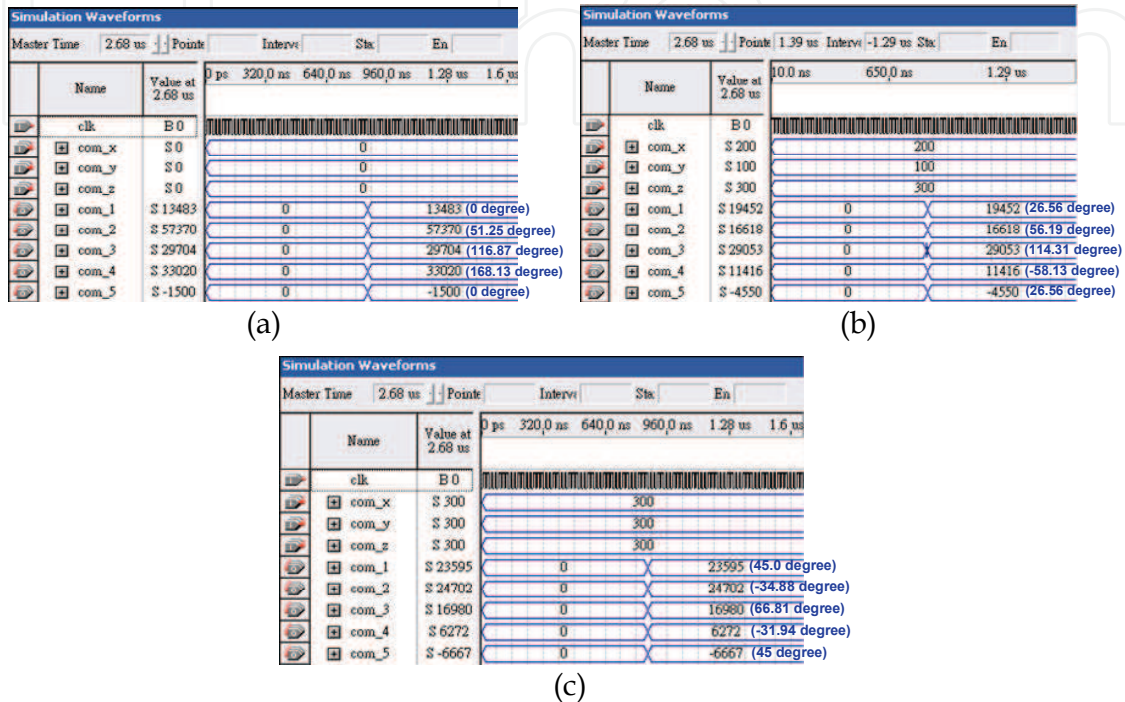

(a)   (b)

(c)

Figure 11. Simulation result of Cartesian space (a) (0,0,0) (b) (200,100,300) (c) (300,300,300) to join space

Module Circuit	Resource usage in FPGA (ALUTs)
Nios II embedded processor IP	2,468
Five sets of PMW module	316
Five sets of QEP module	285
Position controller module	206
Speed controller module	748
Inverse kinematics module	5,322
Others circuit	71
Total	**9,416**

Table 3. Utility evaluation of the motion control IC for robot manipulator in FPGA

5. Experimental system and results

Figure 12 presents the overall experimental system which includes an FPGA experimental board, five sets of inverter, five sets rectifier, a power supplier system and a Mitsubishi Movemaster RV-M1 micro articulated robot. The micro articulated robot has five servo axes

(excluding the hand) and its specification is shown in Fig.13. Each axis is driven by a 24V DC servo motor with a reduction gear. The operation ranges of the articulated robot are wrist roll ±180 degrees (J5-axis), wrist pitch ±90 degrees (J4-axis), elbow rotation 110 degrees (J3-axis), shoulder rotation 130 degrees (J2-axis) and waist rotation 300 degrees (J1-axis). The gear ratios for J1 to J5 axis of the robot are 1:100, 1:170, 1:110, 1:180 and 1:110, respectively. Each DC motor is attached an optical encoder. Through four times frequency circuit, the encoder pulses generate 800pulses/cycle at J1 to J3 axis and 380pulses/cycle at J4 and J5 axis. The maximum path velocity is 1000mm/s and the lifting capacity is 1.2kg including the hand. The total weight of this robot is 19 kg. The inverter has 4 sets of IGBT type power transistors. The collector-emitter voltage of the IGBT is rating 600V, the gate-emitter voltage is rating ±12V, and the collector current in DC is rating 25A and in short time (1ms) is 50A. The photo-IC, Toshiba TLP250, is used for gate driving circuit of IGBT. Input signals of the inverter are PWM signals from FPGA chip. The FPGA-Altera Stratix II EP2S60F672C5ES in Fig. 1(a) is used to develop a full digital motion controller for robot manipulator. A Nios II embedded processor can be download to this FPGA chip.
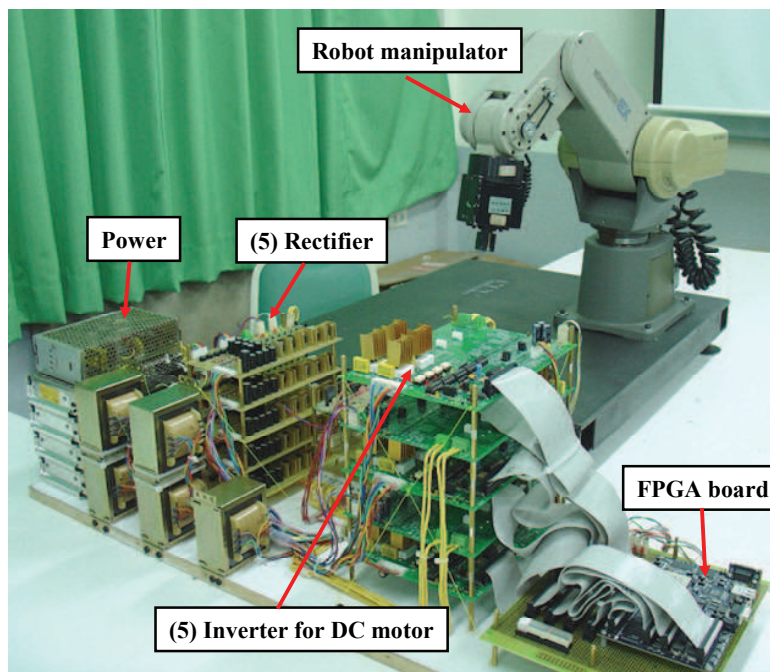

Figure 12. Experimental system

In Fig.12 or Fig.4, the realization in PWM switching frequency, dead-band of inverter, position and speed control sampling frequency are set at 18k Hz, 1.28μs, 762 Hz and 1525 Hz, respectively. Moreover, in the position loop P controller design, the controller parameters at each axis of robot manipulator are selected with identical values by P-gain with 2.4. However, in the speed loop PI controller design, the controller parameters at each J1~J5 axis are selected with different values by [3.17, 0.05], [3.05, 0.12], [2.68, 0.07], [2.68, 0.12] and [2.44, 0.06], respectively.

To confirm the effectiveness of the proposed motion control IC, the square-wave position command with ±3 degrees amplitude and 2 seconds period is firstly adopted to test the dynamic response performance. At the beginning of the step response testing, the robot manipulator is moved to a specified attitude for joints J1-J5 rotating at the [9°, 40°, 60°, 45°, 10°] position. Figure 14 shows the experimental results of the step response under these design

conditions, where the rise time of the step responses are with 124ms, 81ms, 80ms, 151ms and 127 ms for axis J1-J5, respectively. The results also indicate that these step responses have almost zero steady-state error and no oscillation. Next, to test the performance of a point-to-point motion control for the robot manipulator, a specified path is run where the robot moves from the point 1 position, (94.3, 303.5, 403.9) mm to the point 2 position, (299.8, 0, 199.6) mm, then back to point 1. After through inverse kinematics computation in (17) ~ (23), moving each joint rotation angle of the robot from point 1 to point 2 need rotation of -72.74o (-16,346 Pulses), 23.5o (8,916 Pulses), 32.16o (8,173 Pulses), -56.72o (-11,145 Pulses) and -71.18o (-8,173 Pulses), respectively. Additionally, a point-to-point control scheme with constant acceleration/deceleration trapezoid velocity profile adopts to smooth the robot manipulator movement. Applying this motion control scheme, all joins of robot rotate with simultaneous starting and simultaneous stopping time. The acceleration/deceleration time and overall running time are set to 256ms and 1s, and the computation procedure in paragraph 3.3.1 is applied. Figure 15 shows the tracking results of using this design condition at each link. The results indicate that the motion of each robot link produces perfect tracking with the target command in the position or the velocity response. Furthermore, the path trajectory among the position command, actual position trajectory and point-to-point straight line in Cartesian space R^3 (x,y,z) are compared, with the results shown in Fig. 16. Analytical results indicate that the actual position trajectory can precisely track the position command, but that the middle path between two points can not be specified in advanced. Next, the performance of the linear trajectory tracking of the proposed motion control IC is tested, revealing that the robot can be precisely controlled at the specified path trajectory or not. The linear trajectory path is specified where the robot manipulator moves from the starting position, (200, 100, 400) mm to the ending position, (300, 0, 300) mm, then back to starting position. The linear trajectory command is generated 100 equidistant segmented points from the starting to the ending position. Each middle position at Cartesian space R^3 (x,y,z) will be transformed to joint space $(\theta_1^*,\theta_2^*,\theta_3^*,\theta_4^*,\theta_5^*)$, through the inverse kinematics computation, then sent to the servo controller of the robot manipulator. The tracking result of linear trajectory tracking is displayed in Fig. 17. The overall running time is 1 second and the tracking errors are less than 4mm. Similarly, the circular trajectory command generated by 300 equidistant segmented points with center (220,200,300) mm and radius 50 mm is tested again and its tracking result is shown in Fig. 18. The overall running time is 3 second and the trajectory tracking errors at each axis are less than 2.5mm. Figures 17~18 show the good motion tracking results under a prescribed planning trajectory. Therefore, experimental results from Figs. 14~18, demonstrate that the proposed FPGA-based motion control IC for robot manipulator ensures effectiveness and correctness

Movemaster RV-M1 micro articulated robot

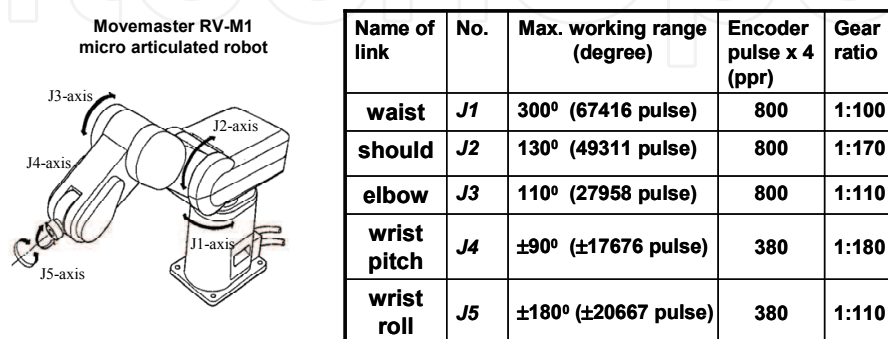| Name of link	No.	Max. working range (degree)	Encoder pulse x 4 (ppr)	Gear ratio
waist	J1	300⁰ (67416 pulse)	800	1:100
should	J2	130⁰ (49311 pulse)	800	1:170
elbow	J3	110⁰ (27958 pulse)	800	1:110
wrist pitch	J4	±90⁰ (±17676 pulse)	380	1:180
wrist roll	J5	±180⁰ (±20667 pulse)	380	1:110

Figure 13. Mitsubishi Movemaster RV-M1 micro articulated robot
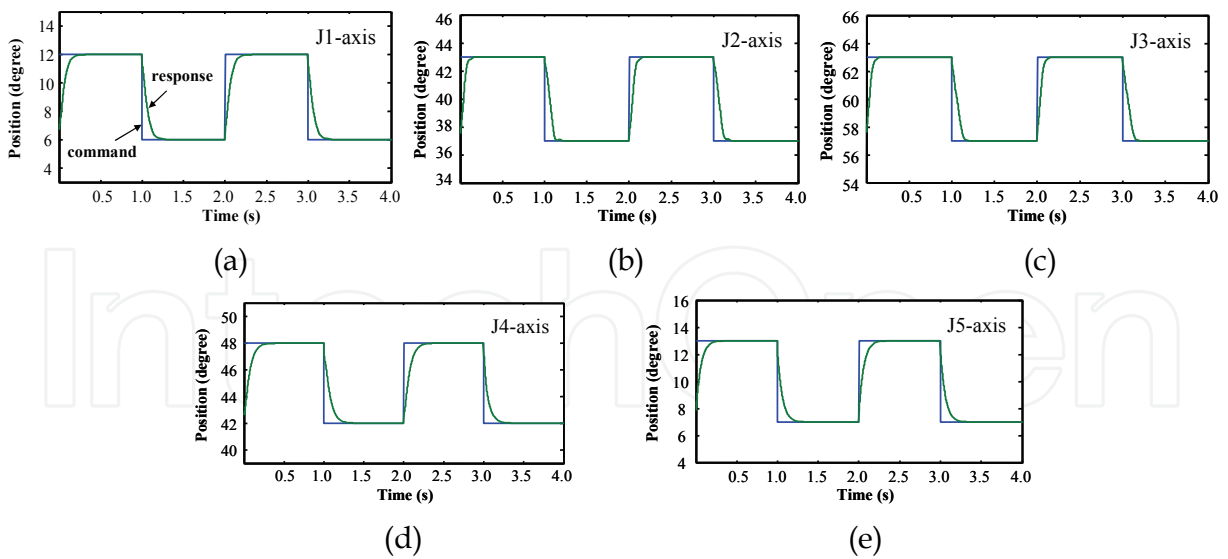
Figure 14 Position step response of robot manipulator at (a) J1 axis (b) J2 axis (c) J3 axis (d) J4 axis (e) J5 axis
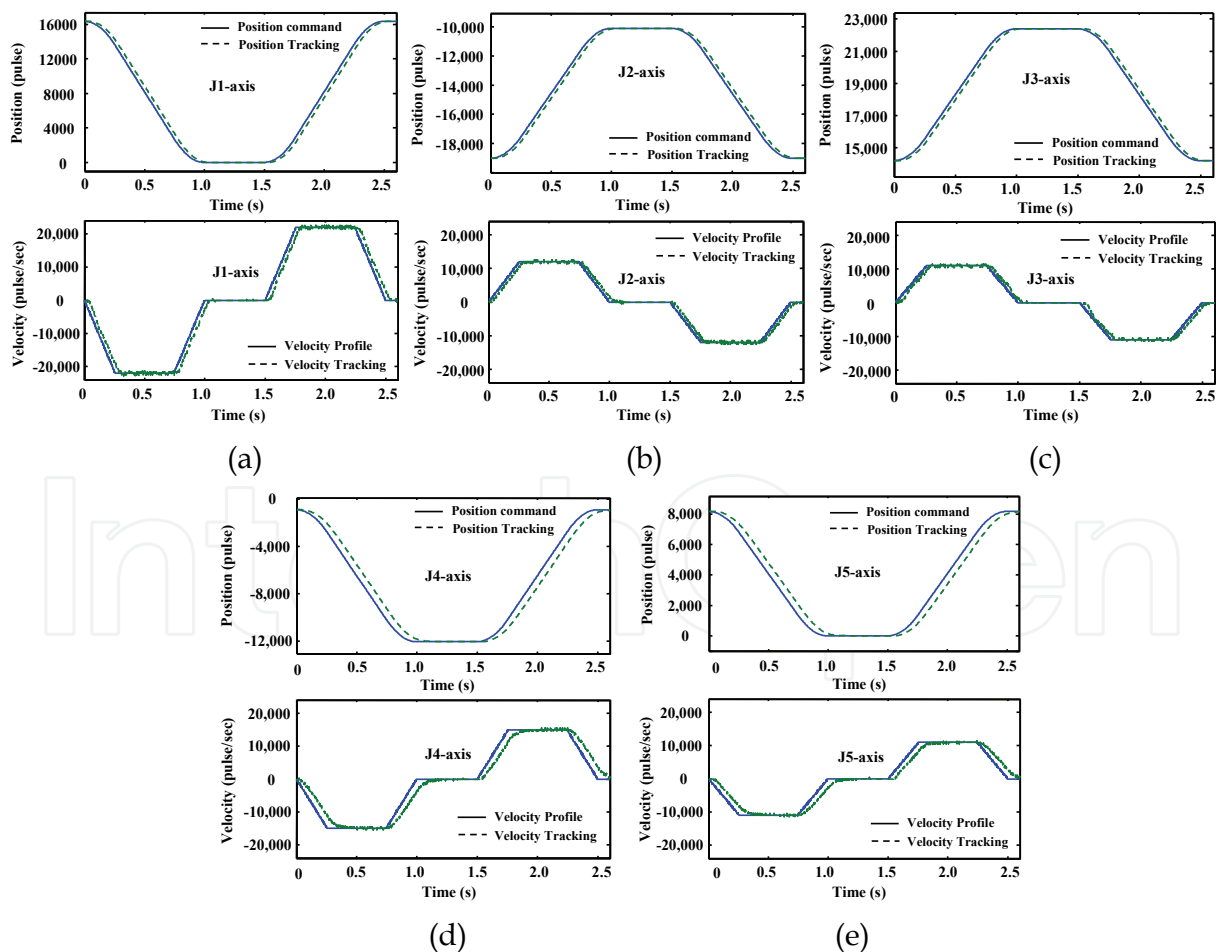

Figure 15 Position and its velocity profile tracking response of robot manipulator at (a) J1 axis (b) J2 axis (c) J3 axis (d) J4 axis (e) J5 axis
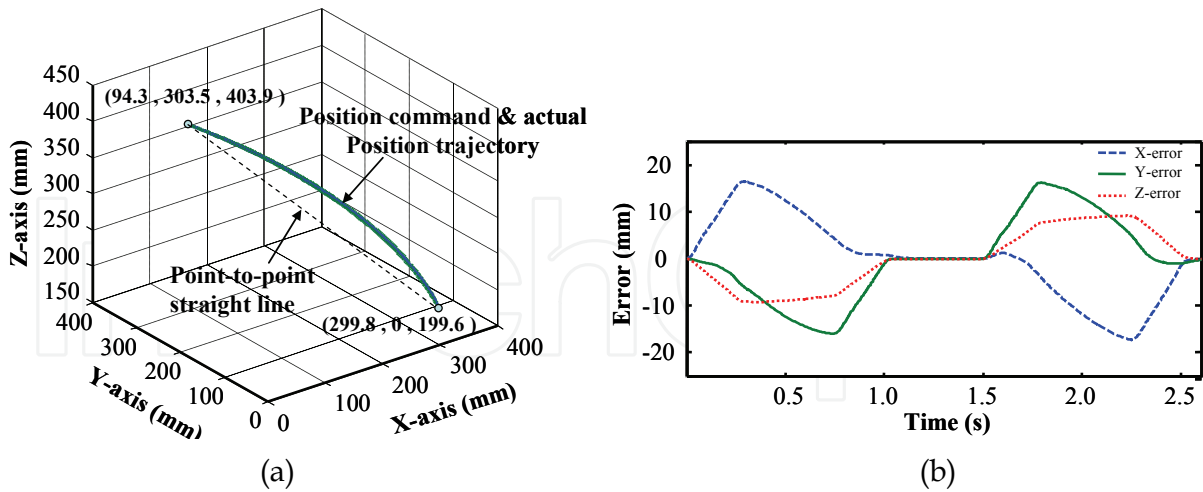
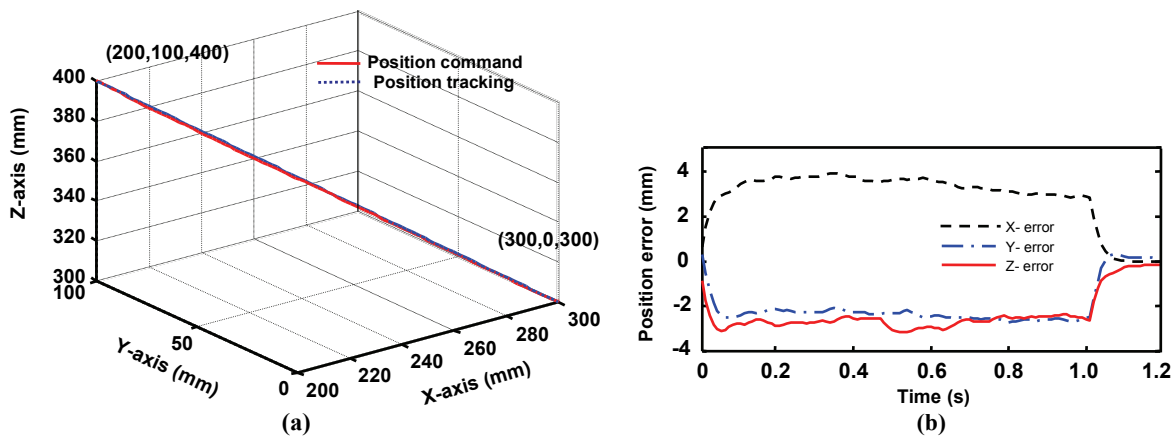Figure 16 (a) Point-to-point path tracking result of robot manipulator (b) tracking error

Fig. 17 (a) Linear trajectory tracking result of robot manipulator (b) tracking error
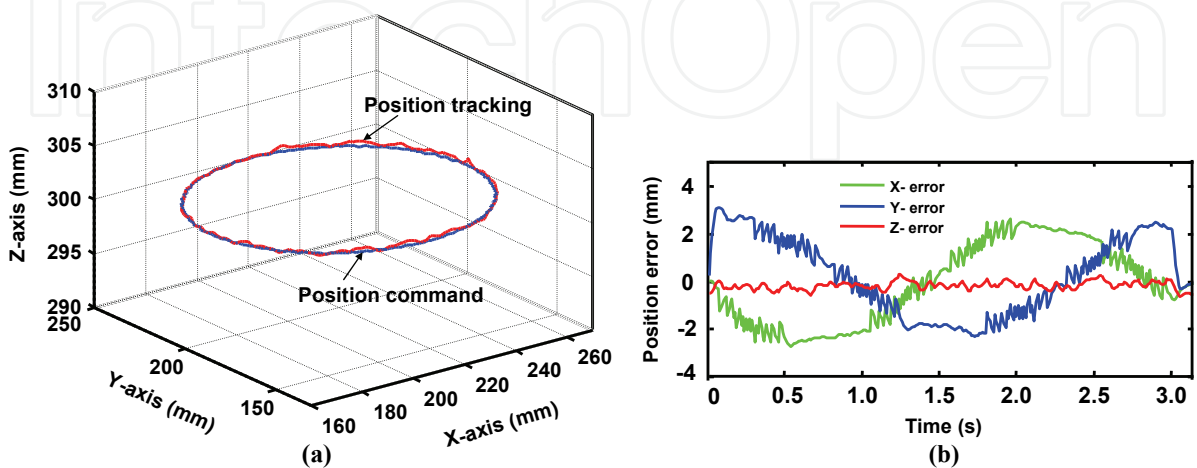

Figure 18 (a) Circular trajectory tracking result of robot manipulator (b) tracking error

6. Conclusion

This study presents a motion control IC for robot manipulator based on novel FPGA technology. The main contributions herein are summarized as follows.

1. The functionalities required to build a fully digital motion controller of a five-axis robot manipulator, such as the function of a motion trajectory planning, an inverse kinematics, five axes position and speed controller, five sets of PWM and five sets of QEP circuits have been integrated and realized in one FPGA chip.

2. The function of inverse kinematics is successfully implemented by hardware in FPGA; as the result, it diminishes the computation time from 5.6ms using Nios II processor to 840ns using FPGA hardware, and increases the system performance.

3. The software/hardware co-design technology under SoPC environment has been successfully applied to the motion controller of robot manipulator.

Finally, the experimental results by the step response, the point-to-point motion trajectory response and the linear and circular motion trajectory response, have been revealed that based on the novel FPGA technology, the software/hardware co-design method with parallel operation ensures a good performance in the motion control system of robot manipulator.

Compared with DSP, using FPGA in the proposed control architecture has the following benefits.

1. Inverse kinematics and servo position controllers are implemented by hardware and the trajectory planning is implemented by software, which can all be programmable design. Therefore, the flexibility of designing a specified function of robot motion controller is greatly increased.

2. Parallel processing in each block function of the motion controller makes the dynamic performance of the robot's servo drive increasable.

3. In the commercial DSP product, it is difficult to integrate all the functions of implementing a five-axis motion controller for robot manipulator into only one chip.

7. References

Altera Corporation, (2004). *SOPC World*.

Altera (2008): www.altera.com

Hall, T.S. & Hamblen, J.O. (2004). System-on-a-programmable-chip development platforms in the classroom, *IEEE Trans. on Education*, Vol. 47, No. 4, pp.502-507.

Monmasson, E. & Cirstea, M.N. (2007) FPGA design methodology for industrial control systems – a review, *IEEE Trans. Industrial Electronics*, Vol. 54, No. 4, pp.1824-1842.

Kabuka, M.; Glaskowsky, P. & Miranda, J. (1988). Microcontroller-based Architecture for Control of a Six Joints Robot Arm, *IEEE Trans. on Industrial Electronics*, Vol. 35, No. 2, pp. 217-221.

Kung, Y.S. & Shu, G.S. (2005). Design and Implementation of a Control IC for Vertical Articulated Robot Arm using SOPC Technology, *Proceeding of IEEE International Conference on Mechatronics*, 2005, pp. 532~536.

Kung, Y.S.; Tseng, K.H. & Tai, F.Y. (2006) FPGA-based servo control IC for X-Y table, *Proceedings of the IEEE International Conference on Industrial Technology*, pp. 2913-2918.

Kung, Y.S. & Tsai, M.H. (2007). FPGA-based speed control IC for PMSM drive with adaptive fuzzy control, *IEEE Trans. on Power Electronics*, Vol. 22, No. 6, pp. 2476-2486.

Lewis, F.L.; Abdallah C.T. & Dawson, D.M. (1993). *Control of Robot Manipulators,* Macmillan Publishing Company.

Li, T.S.; Chang S.J. & Chen, Y.X. (2003) Implementation of Human-like Driving Skills by Autonomous Fuzzy Behavior Control on an FPGA-based Car-like Mobile Robot, *IEEE Trans. on Industrial Electronics*, Vol. 50, No.5, pp. 867-880.

Oh, S.N.; Kim, K.I. & Lim, S. (2003). Motion Control of Biped Robots using a Single-Chip Drive, *Proceeding of IEEE International Conference on Robotics & Automation*, pp. 2461~2469.

Schilling, R. (1998). *Fundamentals of Robotics – Analysis and control,* Prentice-Hall International.

Shao, X. & Sun, D. (2005). A FPGA-based Motion Control IC Design, *Proceeding of IEEE International Conference on Industrial Technology*, pp. 131-136.

Wei, R.; Gao, X.H.; Jin, M.H.; Liu, Y.W.; Liu, H.; Seitz, N.; Gruber, R. & Hirzinger, G. (2005). FPGA based Hardware Architecture for HIT/DLR Hand, *Proceeding of IEEE/RSJ International Conference on intelligent Robots and System*, pp. 523~528.

Xu, N.; Liu, H.; Chen, X. & Zhou, Z. (2003). Implementation of DVB Demultiplexer System with System-on-a-programmable-chip FPGA, *Proceeding of 5th International Conference on ASIC*, Vol. 2, pp. 954-957.

Yang, G.; Liu, Y.; Cui, N. & Zhao, P. (2006). Design and Implementation of a FPGA-based AC Servo System, *Proceedings of the Sixth World Congress on the Intelligent Control and Automation*, Vol. 2, pp. 8145-8149.

Yasuda, G. (2000). Microcontroller Implementation for Distributed Motion Control of Mobile Robots, *Proceeding of International workshop on Advanced Motion Control,* pp. 114-119.

Zhou, Z.; Li, T.; Takahahi, T. & Ho, E. (2004). FPGA realization of a high-performance servo controller for PMSM, *Proceeding of the 9th IEEE Application Power Electronics conference and Exposition,* Vol.3, pp. 1604-1609.

Robot Manipulators

Edited by Marco Ceccarelli

ISBN 978-953-7619-06-0

Hard cover, 546 pages

Publisher InTech

Published online 01, September, 2008

Published in print edition September, 2008

In this book we have grouped contributions in 28 chapters from several authors all around the world on the several aspects and challenges of research and applications of robots with the aim to show the recent advances and problems that still need to be considered for future improvements of robot success in worldwide frames. Each chapter addresses a specific area of modeling, design, and application of robots but with an eye to give an integrated view of what make a robot a unique modern system for many different uses and future potential applications. Main attention has been focused on design issues as thought challenging for improving capabilities and further possibilities of robots for new and old applications, as seen from today technologies and research programs. Thus, great attention has been addressed to control aspects that are strongly evolving also as function of the improvements in robot modeling, sensors, servo-power systems, and informatics. But even other aspects are considered as of fundamental challenge both in design and use of robots with improved performance and capabilities, like for example kinematic design, dynamics, vision integration.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ying-Shieh Kung and Chia-Sheng Chen (2008). FPGA-Realization of a Motion Control IC for Robot Manipulator, Robot Manipulators, Marco Ceccarelli (Ed.), ISBN: 978-953-7619-06-0, InTech, Available from: http://www.intechopen.com/books/robot_manipulators/fpga-realization_of_a_motion_control_ic_for_robot_manipulator

INTECH

open science | open minds