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

21

FPGA-Realization of a Motion Control IC for X-Y Table

Ying-Shieh Kung and Ting-Yu Tai
Southern Taiwan University
Taiwan

1. Introduction

The development of a compact and high performance motion controller for precision X-Y table, CNC machine etc. has been a popular field in literature (Goto et al., 1996; Wang & Lee, 1999; Hanafi et al., 2003). In position control of X-Y table, there are two approaches to be considered. One is semi closed-loop control and the other is full closed-loop control. The full closed-loop control with feed-backed by a linear encoder as the table position signal has a better positioning performance than the semi closed-loop control that a rotary encoder attached to PMSM is feed-backed as the position signal. However, to develop a motion control IC for X-Y table, the fixed-point digital signal processor (DSP) and FPGA provide two possible solutions in this issue. Compared with FPGA, DSP suffers from a long period of development and exhausts many resources of the CPU (Zhou et al., 2004).

For the progress of VLSI technology, the FPGA has been widely investigated due to its programmable hard-wired feature, fast time-to-market, shorter design cycle, embedding processor, low power consumption and higher density for implementing digital control system (Monmasson & Cirstea, 2007; Naouar et al., 2007; Jung & Kim, 2007). FPGA provides a compromise between the special-purpose ASIC (application specified integrated circuit) hardware and general-purpose processors (Wei et al., 2005). Therefore, using an FPGA to form a compact, low-cast and high performance servo system for precision machine has become an important issue. However, in many researches, the FPGA is merely used to realize the hardware part of the overall control system. Recently, fuzzy control has been successfully demonstrated in industrial control field (Sanchez-Solano et al., 2007; Kung & Tsai, 2007).Compared with other nonlinear approaches, FC has two main advantages, as follows: (1) FC has a special non-linear structure that is universal for various or uncertainty plants. (2) the formulation of fuzzy control rule can be easily achieved by control engineering knowledge, such as dynamic response characteristics, and it doesn't require a mathematical model of controlled plant. In literature, Li et al. (2003) utilized an FPGA to implement autonomous fuzzy behavior control on mobile robot. Lin et al. (2005) presented a fuzzy sliding-mode control for a linear induction motor drive based on FPGA. But, due to the fuzzy inference mechanism module adopts parallel processing circuits, it consumes much more FPGA resources; therefore limited fuzzy rules are used in their proposed method. To solve this problem, a FSM joined by a multiplier, an adder, a LUT (Look-up table), some comparators and registers are proposed to model the FC algorithm of the

PMSM drive system. Then a VHDL is adopted to describe the circuit of the FSM (Hsu et al., 1996). Due to the FSM belongs to the sequential processing method; the FPGA resources usage can be greatly reduced. Further, in recent years, an embedded processor IP and an application IP can now be developed and downloaded into FPGA to construct a SoPC environment (Altera, 2004), allowing the users to design a SoPC module by mixing hardware and software in one FPGA chip (Hall & Hamblen, 2004). The circuits required fast processing but fixed computation are suitable to be implemented by hardware in FPGA, and the heavy computation or complicated processing can be realized by software in FPGA (Kung et al., 2004; Kung & Shu, 2005). The results of the software/hardware co-design increase the programmability, flexibility of the designed digital system, enhance the system performance by parallel processing and reduce the development time.

To exploit the advantages, a motion control IC for X-Y table based on the new-generation FPGA technology is developed in this study and shown in Fig.1 (Kung et al., 2006), which the scheme of position/speed/current vector control of two PMSMs can be realized by hardware in FPGA, and the motion trajectory for X-Y table can be realized by software using Nios II embedded processor. Hence, all functionalities, which are based on software/hardware co-design, required to construct a full closed-loop control for X-Y table can be integrated and implemented in one FPGA chip. In addition, the FPGA resources usage can be greatly reduced by using the FSM in the control algorithm design. Herein, the 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, are used. Finally, an experimental system included by an FPGA experimental board, two inverters, two sets of A/D converter and an X-Y table, is set up to verify the correctness and effectiveness of the proposed FPGA-based motion control IC.

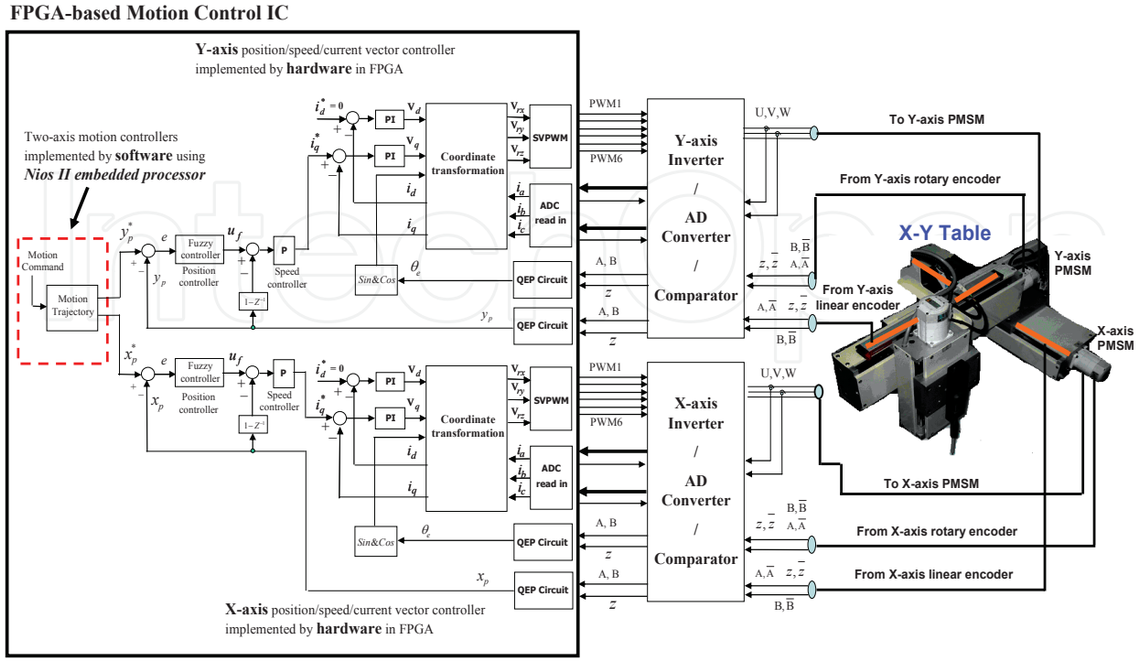


Fig. 1. The architecture of the FPGA-based motion control system for X-Y table

2. System description and controller design of X-Y table

The X-Y table is driven by two PMSMs which the current, speed and position loop in each PMSM drive adopts vector control, P control and fuzzy control, respectively The architecture of the proposed FPGA-based motion control IC for X-Y table is shown in Fig. 1. The modeling of PMSM, the fuzzy control algorithm and the motion trajectory planning are introduced as follows:

2.1 Mathematical model of PMSM and current vector controller

The typical mathematical model of a PMSM is described, in two-axis d-q synchronous rotating reference frame, as follows

$$\frac{di_d}{dt} = -\frac{R_s}{L_d} i_d + \omega_e \frac{L_q}{L_d} i_q + \frac{1}{L_d} v_d \tag{1}$$

$$\frac{di_q}{dt} = -\omega_e \frac{L_d}{L_q} i_d - \frac{R_s}{L_q} i_q - \omega_e \frac{\lambda_f}{L_q} + \frac{1}{L_q} v_q \tag{2}$$

where v_d, v_q are the d and q axis voltages; i_d, i_q, are the d and q axis currents, R_s is the phase winding resistance; L_d, L_q are the d and q axis inductance; ω_e is the rotating speed of magnet flux; λ_f is the permanent magnet flux linkage.

The current loop control of PMSM drive in Fig.1 is based on a vector control approach. That is, if the i_d is controlled to 0 in Fig.1, the PMSM will be decoupled and controlling a PMSM like to control a DC motor. Therefore, after decoupling, the torque of PMSM can be written as the following equation,

$$T_e = \frac{3P}{4} \lambda_f i_q \triangleq K_t i_q \tag{3}$$

with

$$K_t = \frac{3P}{4} \lambda_f \tag{4}$$

Finally, considering the mechanical load with linear table, the overall dynamic equation of linear table system is obtained by

$$T_e - T_L = J_m \frac{2\pi}{r} \frac{d^2 s_p}{dt^2} + B_m \frac{2\pi}{r} \frac{ds_p}{dt} \tag{5}$$

where T_e is the motor torque, K_t is force constant, J_m is the inertial value, B_m is damping ratio, T_L is the external torque, s_p represents the displacement of X-axis or Y-axis table and r is the lead of the ball screw.

The current loop of the PMSM drive for X- or Y-table in Fig.1 includes two PI controllers, coordinate transformations of Clark, Modified inverse Clark, Park, inverse Park, SVPWM (Space Vector Pulse Width Muldulation), pulse signal detection of the encoder etc. The coordination transformation of the PMSM in Fig. 1 can be described in synchronous rotating reference frame. Figure 2 is the coordination system in rotating motor which includes

stationary *a-b-c* frame, stationary α-β frame and synchronously rotating *d-q* frame. Further, the formulations among three coordination systems are presented as follows.

1. *Clarke*: stationary *a-b-c* frame to stationary α-β frame.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{6}$$

2. Modified $Clarke^{-1}$: stationary α-β frame to stationary *a-b-c* frame.

$$\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_\beta \\ v_\alpha \end{bmatrix} \tag{7}$$

3. *Park* : stationary α-β frame to rotating *d-q* frame.

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{8}$$

4. $Park^{-1}$: rotating *d-q* frame to stationary α-β frame.

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix} \tag{9}$$

where θ_e is the electrical angle.

In Fig. 1, two digital *PI* controllers are presented in the current loop of PMSM. For the example in *d* frame, the formulation is shown as follows.

$$e_d(k) = i_d^*(k) - i_d(k) \tag{10}$$

$$v_{p\_d}(k) = k_{p\_d}\, e_d(k) \tag{11}$$

$$v_{i\_d}(k) = v_{i\_d}(k-1) + k_{i\_d}\, e_d(k-1) \tag{12}$$

$$v_d(k) = v_{p\_d}(k) + v_{i\_d}(k) \tag{13}$$

the e_d is the error between current command and measured current. The k_{p_d}, k_{i_d} are *P* controller gain and *I* controller gain, respectively. The $v_{p\_d}(k), v_{i\_d}(k), v_d(k)$ are the output of *P* controller only, *I* controller only and the *PI* controller, respectively. Similarity, the formulation of *PI* controller in *q* frame is the same.

2.2 Fuzzy controller (FC) for position control loop

The position controllers in X-axis and Y-axis table of Fig. 1 adopt fuzzy controller, which includes fuzzification, fuzzy rules, inference mechanism and defuzzification. Herein, an FC design method for X-axis and Y-axis table is presented. At first, position error and its error change, e , de are defined by

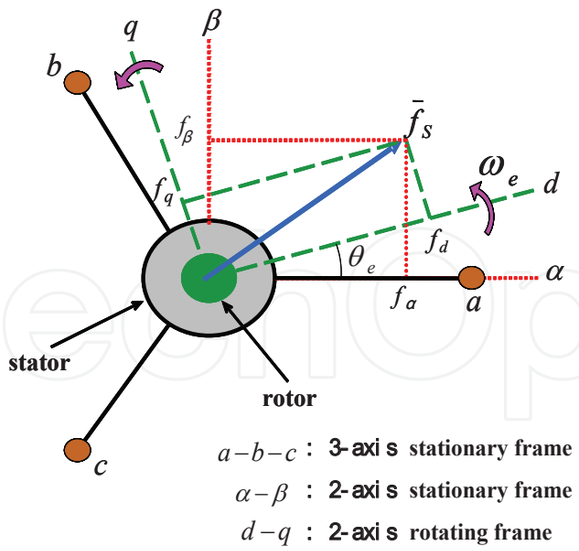


Fig. 2. Transformation between stationary axes and rotating axes

$$e(k) = s_p^*(k) - s_p(k) \tag{14}$$

$$de(k) = e(k) - e(k-1) \tag{15}$$

The K_{er} and K_{der} are the gains of the input variables e and de, respectively, as well as u_f is the output variables of the FC. The design procedure of the FC is as follows:

a. Take the E and dE as the input linguist variables, which are defined by {A_0, A_1, A_2, A_3, A_4, A_5, A_6} and {B_0, B_1, B_2, B_3, B_4, B_5, B_6}, respectively. Each linguist value of E and dE are based on the symmetrical triangular membership function which is shown in Fig.3. The symmetrical triangular membership function are determined uniquely by three real numbers $\xi_1 \le \xi_2 \le \xi_3$, if one fixes $f(\xi_1) = f(\xi_3) = 0$ and $f(\xi_2) = 1$. With respect to the universe of discourse of [-6.6], the numbers for these linguistic values are selected as follows:

$$A_0{=}B_0: \{-6,-6,-4\},\ A_1{=}B_1: \{-6,-4,-2\},\ A_2{=}B_2: \{-4,-2,0\},\ A_3{=}B_3: \{-2,0,2\},$$
$$A_4{=}B_4: \{0,2,4\},\ A_5{=}B_5: \{2,4,6\},\ A_6{=}B_6: \{4,6,6\} \tag{16}$$

b. Compute the membership degree of e and de. Figure 3 shows that the only two linguistic values are excited (resulting in a non-zero membership) in any input value, and the membership degree $\mu_{A_i(e)}$ can be derived, in which the error e is located between e_i and e_{i+1}, two linguist values of A_i and A_{i+1} are excited, and the membership degree is obtained by

$$\mu_{A_i}(e) = \frac{e_{i+1} - e}{2} \quad \text{and} \quad \mu_{A_{i+1}}(e) = 1 - \mu_{A_i}(e) \tag{17}$$

where $e_{i+1} \triangleq -6 + 2*(i+1)$. Similar results can be obtained in computing the membership degree $\mu_{B_j(de)}$.

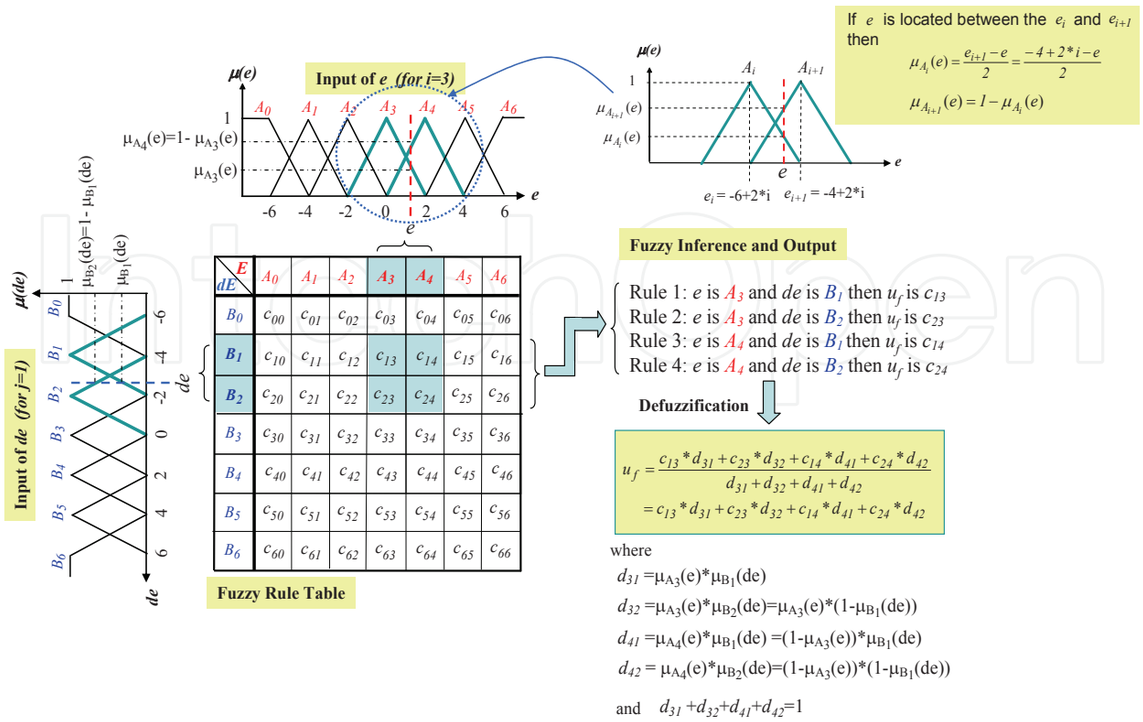


Fig. 3. Fuzzification, fuzzy rule table, fuzzy inference and defuzzication

c. Select the initial fuzzy control rules by referring to the dynamic response characteristics (Liaw et al., 1999), such as,

$$IF\ \ e\ is\ A_i\ \ and\ \ \Delta e\ is\ B_j\ \ THEN\ \ u_f\ \ is\ \ c_{j,i} \tag{18}$$

where i and j = 0~6, A_i and B_j are fuzzy number, and $c_{j,i}$ is real number. The graph of fuzzification and fuzzy rule table is shown in Fig. 3.

d. Construct the fuzzy system $u_f(e,de)$ by using the singleton fuzzifier, product-inference rule, and central average defuzzifier method. Although there are total 49 fuzzy rules in Fig. 3 will be inferred, actually only 4 fuzzy rules can be effectively excited to generate a non-zero output. Therefore, if the error e is located between e_i and e_{i+1}, and the error change de is located between de_j and de_{j+1}, only four linguistic values A_i, A_{i+1}, B_j, B_{j+1} and corresponding consequent values $c_{j,i}$, $c_{j+1,i}$, $c_{j,i+1}$, $c_{j+1,i+1}$ can be excited, and the (18) can be replaced by the following expression:

$$u_f(e,de)=\frac{\sum_{n=i}^{i+1}\sum_{m=j}^{j+1}c_{m,n}[\mu_{A_n}(e)*\mu_{B_m}(de)]}{\sum_{n=i}^{i+1}\sum_{m=j}^{j+1}\mu_{A_n}(e)*\mu_{B_m}(de)}\triangleq\sum_{n=i}^{i+1}\sum_{m=j}^{j+1}c_{m,n}*d_{n,m} \tag{19}$$

where $d_{n,m}\triangleq\mu_{A_n}(e)*\mu_{B_m}(de)$. And those $c_{m,n}$ denote the consequent parameters of the fuzzy system.

2.3 Motion trajectory planning of X-Y table

The point-to-point, circular and window motion trajectories are usually considered to evaluate the motion performance for X-Y table.

a. In point-to-point motion trajectory, for smoothly running of the table, it is designed with the trapezoidal velocity profile and its formulation is shown as follows.

$$s(t)=\begin{cases}\frac{1}{2}At^2+s_0 & 0\le t\le t_a\\ v_m(t-t_a)+s(t_a) & t_a\le t\le t_d\\ -\frac{1}{2}A(t-t_d)^2+v_m(t-t_d)+s(t_d) & t_d\le t\le t_s\end{cases} \tag{20}$$

Where $0<t<t_a$ is at the acceleration region, $t_a<t<t_d$ is at the constant velocity region, and $t_d<t<t_s$ is at the deceleration region. The s represents the position command in X-axis or Y-axis table; A is the acceleration/deceleration value; s_0 is the initial position; v_m is the maximum velocity; t_a, t_d and t_s represents the end time of the acceleration region, the start time of the deceleration region and the end time of the trapezoidal motion, respectively.

b. In circular motion trajectory, it is computed by

$$x_i = r\,sin(\theta_i) \tag{21}$$

$$y_i = r\,cos(\theta_i) \tag{22}$$

with $\theta_i=\theta_{i-1}+\Delta\theta$. Where $\Delta\theta$, r, x_i, y_i are angle increment, radius, X-axis trajectory command and Y-axis trajectory command, respectively.

c. The window motion trajectory is shown in Fig.4. The formulation is derived as follows:

$$\text{a-trajectory}: x_i=x_{i-1}, y_i=S+y_{i-1} \tag{23}$$

$$\text{b-trajectory}: (\theta_i:\frac{6}{4}\pi\to 2\pi, \text{and } \theta_i=\theta_{i-1}+\Delta\theta)$$

$$x_i=O_{x1}+r\cos(\theta_i), y_i=O_{y1}+r\sin(\theta_i) \tag{24}$$

$$\text{c-trajectory}: x_i=S+x_{i-1}, y_i=y_{i-1} \tag{25}$$

$$\text{d-trajectory}: (\theta_i:\pi\to\frac{6}{4}\pi, \text{and } \theta_i=\theta_{i-1}+\Delta\theta)$$

$$x_i=O_{x2}+r\cos(\theta_i), y_i=O_{y2}+r\sin(\theta_i) \tag{26}$$

$$\text{e-trajectory}: x_i=x_{i-1}, y_i=-S+y_{i-1} \tag{27}$$

$$\text{f-trajectory}: (\theta_i:\frac{1}{2}\pi\to\pi, \text{and } \theta_i=\theta_{i-1}+\Delta\theta)$$

$$x_i = O_{x3} + r\cos(\theta_i), y_i = O_{y3} + r\sin(\theta_i) \tag{28}$$

$$\text{g-trajectory} : x_i = -S + x_{i-1}, y_i = y_{i-1} \tag{29}$$

$$\text{h-trajectory} : (\theta_i : 0 \to \frac{1}{2}\pi, \text{and } \theta_i = \theta_{i-1} + \Delta\theta)$$

$$x_i = O_{x4} + r\cos(\theta_i), y_i = O_{y4} + r\sin(\theta_i) \tag{30}$$

$$\text{i-trajectory} : x_i = x_{i-1}, y_i = S + y_{i-1} \tag{31}$$

where S, $\Delta\theta$, x_i, y_i are position increment, angle increment, X-axis trajectory command and Y-axis trajectory command, respectively. In addition, the (O_{x1}, O_{y1}), (O_{x2}, O_{y2}), (O_{x3}, O_{y3}), (O_{x4}, O_{y4}) are arc center of b-, d-, f-, and h-trajectory in the Fig. 4 and r is the radius. The motion speed of the table is determined by $\Delta\theta$.

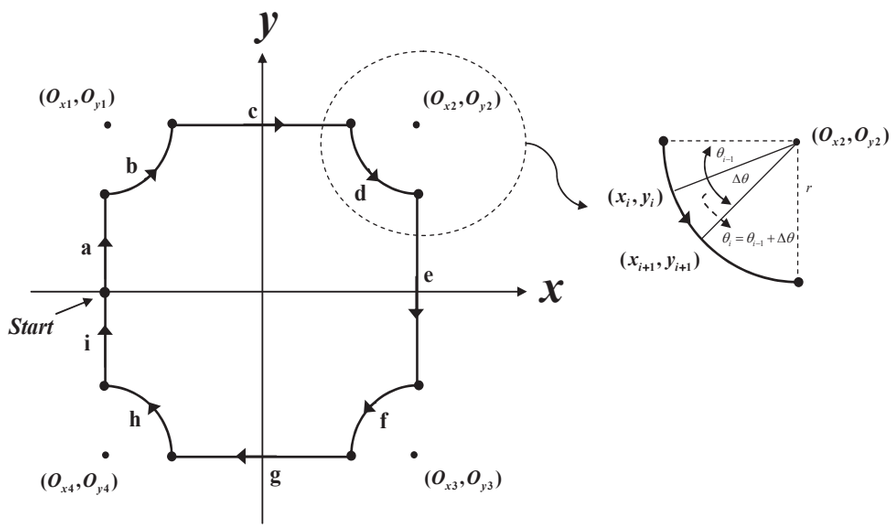


Fig. 4. Window motion trajectory

3. Design of an FPGA-based motion control IC for X-Y table

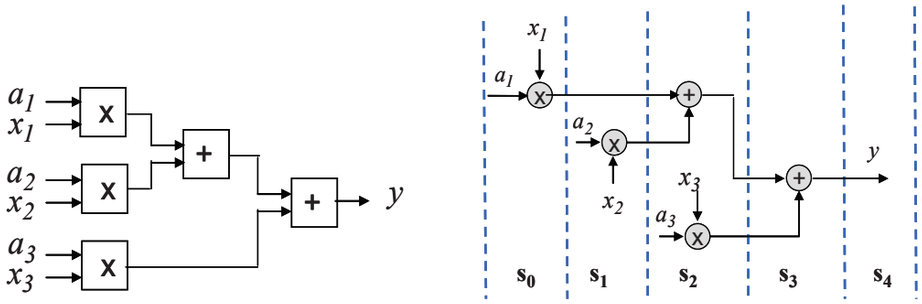
The architecture of the proposed FPGA-based motion control IC for X-Y table is shown in Fig. 1, in which the motion trajectory is implemented by software using Nios II embedded processor and the current vector controller, the position and speed controller for two PMSMs are implemented by hardware in FPGA chip. However, in this section, we firstly introduce the concept of finite state machine (FSM). Then use FSM to design the complicated control algorithm, such as the FC and the vector controller in PMSM drive.

3.1 Finite state machine (FSM)

To reduce the use of the FPGA resource, FSM is adopted to describe the complicated control algorithm. Herein, the computation of a sum of product (SOP) shown below is taken as a case study to present the advantage of FSM.

$$Y = a_1 * x_1 + a_2 * x_2 + a_3 * x_3 \tag{32}$$

Two kinds of design method that one is parallel processing method and the other is FSM method are introduced to realize the the computation of SOP. In the former method, the designed SOP circuit is shown in Fig. 5(a), and it will operate continuously and simultaneously. The circuit needs 2 adders and 3 multipliers, but only one clock time can complete the overall computation. Although the parallel processing method has fast computation ability, it consumes much more FPGA resources. To reduce the resource usage in FPGA, the designed SOP circuit adopted by using the FSM method is proposed and shown in Fig. 5(b), which uses one adder, one multiplier and manipulates 5 steps (or 5 clocks time) machine to carry out the overall computation of SOP. Although the FSM method needs more operation time (if one clock time is 40ns, the 5 clocks needs 0.2 μ s) than the parallel processing method in executing SOP circuit, it doesn't loss any computation power. Therefore, the more complicated computation in algorithm, the more FPGA resources can be economized if the FSM is applied. Further, VHDL code to implement the computation of SOP is shown in Fig.6



(a) (b)

Fig. 5. Computation of SOP by using (a) parallel operation (b) FSM operation

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;
USE IEEE.std_logic_signed.all;
LIBRARY lpm;
USE lpm.LPM_COMPONENTS.ALL;
ENTITY SOP IS
    port(CLK_40n                :IN  STD_LOGIC;
         A1,A2,A3,X1,X2,X3      :IN  STD_LOGIC_VECTOR(11 downto 0);
         Y                      :OUT STD_LOGIC_VECTOR(23 downto 0)
         );
END matrix;
ARCHITECTURE SOP_arch OF SOP IS
    SIGNAL mula,mulb            :STD_LOGIC_VECTOR(11 downto 0);
    SIGNAL mulr                 :STD_LOGIC_VECTOR(23 downto 0);
    SIGNAL adda,addb,addr       :STD_LOGIC_VECTOR(23 downto 0);
    SIGNAL CNT                  :STD_LOGIC_VECTOR(7 downto 0);
BEGIN
multiplier: lpm_mult
  generic map(LPM_WIDTHA=>12,LPM_WIDTHB=>12,LPM_WIDTHS=>12,LPM_WI
              DTHP=>24,LPM_REPRESENTATION=>"signed",LPM_PIPELINE=>1)
  port map(dataa=> mula,datab=> mulb,clock=> clk,result=> mulr);

adder: lpm_add_sub
  generic map(lpm_width=>24,LPM_REPRESENTATION=>"signed",lpm_pipeline=>1)
  port map(dataa=>adda,datab=>addb,clock=> clk,result=>addr);
```

(a)

```
GEN: block
BEGIN
  PROCESS(CLK_40n)
    BEGIN
      IF CLK_40n'EVENT and CLK_40n='1' THEN
        CNT<=CNT+1;
        IF CNT=X"00" THEN
          mula  <= A1;
          mulb  <= X1;
        ELSIF CNT=X"01" THEN
          adda  <= mulr;
          mula  <= A2;
          mulb  <= X2;
        ELSIF CNT=X"02" THEN
          addb  <=  mulr;
          mula  <=  A3;
          mulb  <=  X3;
        ELSIF CNT=X"03" THEN
          adda  <= addr;
          addb  <= mulr;
        ELSIF CNT=X"04" THEN
          Y     <=  addr;
          CNT <=  X"00";
      END IF;
    END IF;
  END PROCESS;
END BLOCK GEN;
END SOP_arch;
```

(a) continue

Fig. 6. VHDL code to implement the computation of SOP

3.2 Design of an FPGA-based motion control IC for X-Y table

The internal architecture of the proposed FPGA-based motion control IC for X-Y table is shown in Fig. 7. The FPGA is used by Altera Stratix II EP2S60 and a Nios II embedded processor can be downloaded into FPGA to construct an SoPC environment. The Altera Stratix II EP2S60 has 48,352 ALUTs (Adaptive Look-UP Tables), maximum 718 user I/O pins, total 2,544,192 RAM bits, and Nios II embedded processor is 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 motion control IC, which is designed in this SoPC environment, comprises a Nios II embedded processor IP and an application IP. The application IP implemented by hardware is adopted to realize two position/speed/current vector controllers of PMSMs and two QEP circuits of linear encoder. The circuit of each current vector controller includes a current controller and coordinate transformation (CCCT), SVPWM generation, QEP detection and transformation, ADC interface, etc. The speed loop uses P controller and the position loop adopts FC. The sampling frequency of the position control loop is designed with 2kHz. The frequency divider generates 50 Mhz (Clk), 25 Mhz (Clk-sp), 16 kHz (Clk-cur), and 2 kHz (Clk-po) clock to supply all circuits in Fig. 7.

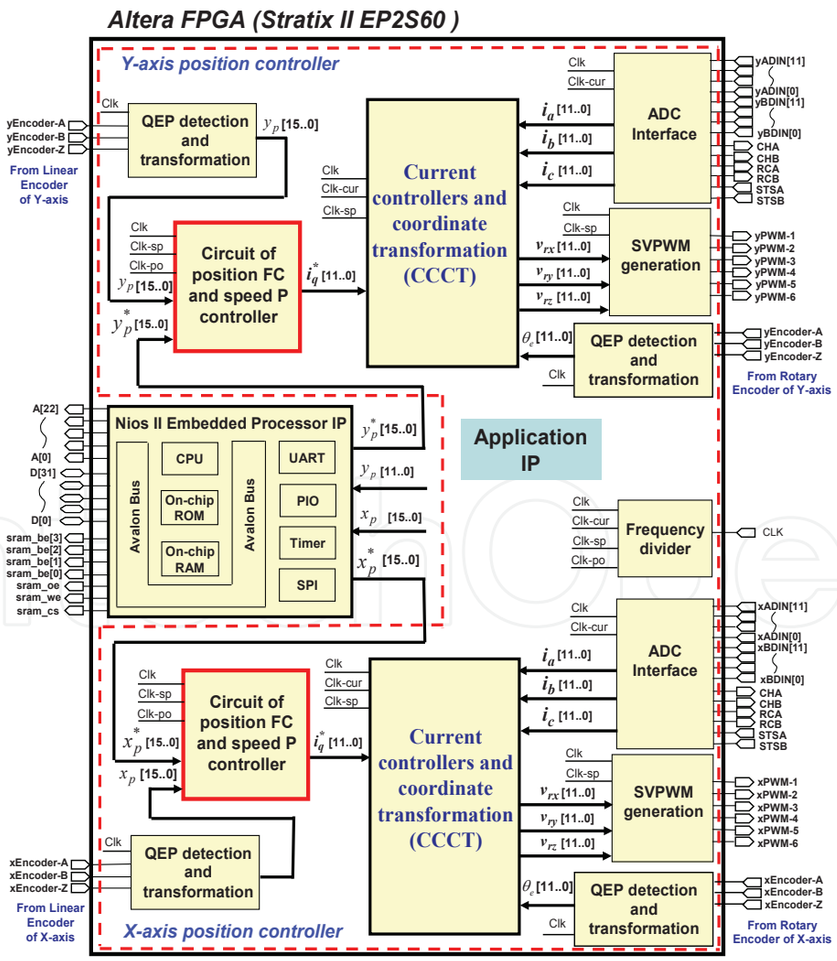


Fig. 7. Internal circuit of the proposed FPGA-based motion control IC

The internal circuit of CCCT performs the function of two PI controllers, table look-up for *sin/cos* function and the coordinate transformation for Clark, Park, inverse Park, modified inverse Clarke. The CCCT circuit designed by FSM is shown in Fig. 8, which uses one adder, one multiplier, an one-bit left shifter, a look-up-table and manipulates 24 steps machine to carry out the overall computation. The data type is 12-bit length with Q11 format and 2's complement operation. In Fig. 8, steps s0~s1 is for the look-up *sin/cos* table; steps s2~s5 and s5~s8 are for the transformation of Clarke and Park, respectively; steps s9~s14 is for the computation of d-axis and q-axis PI controller; and steps s15~s19 and s20~s23 represent the transformation of the inverse Park and the modified inverse Clarke, respectively. The operation of each step in FPGA can be completed within 40ns (25 MHz clock); therefore total 24 steps need 0.96 μs operation time. Although the FSM method needs more operation time than the parallel processing method in executing CCCT circuit, it doesn't loss any control performance in overall system because the 0.96 μs operation time is much less than the designed sampling interval, 62.5 μs (16 kHz) of current control loop in Fig. 1. To prevent numerical overflow and alleviate windup phenomenon, the output values of I controller and PI controller are both limited within a specific range.

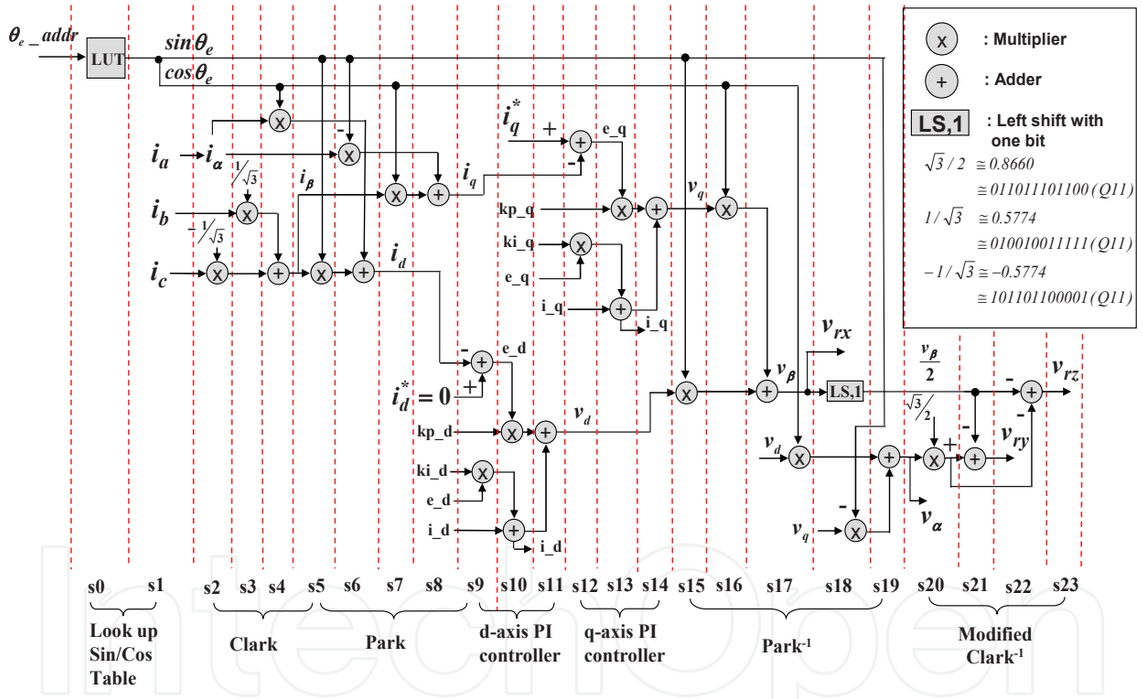


Fig. 8. Designed CCCT circuit in Fig. 7

An FSM is employed to model the FC of the position loop and P controller of the speed loop in PMLSM and shown in Fig. 9, which uses one adder, one multiplier, a look-up table, comparators, registers, etc. and manipulates 23 steps machine to carry out the overall computation. With exception of the data type in reference model are 24-bits, others data type are designed with 12-bits length, 2's complement and Q11 format. Although the algorithm of FC is highly complexity, the FSM can give a very adequate modeling and easily be described by VHDL. Furthermore, steps s_0~s_2 are for the computation of speed, position error and error change; steps s_3~s_6 execute the function of the fuzzification; s_7 describes the look-up table and s_8~s_{16} defuzzification; and steps s_{17}~s_{22} execute the computation of speed

and current command output. The SD is the section determination of *e* and *de*, and its flow chart of circuit design is shown in Fig.10. And the RS,1 represents the right shift function with one bit. The operation of each step in Fig.9 can be completed within 40ns (25 MHz clock) in FPGA; therefore total 23 steps need 0.92μs operation time. It doesn't loss any control performance in the overall system because the operation time with 0.92μs is much less than the sampling interval, 500 μs (2 kHz), of the position control loop in Fig.1.

In Figure 7, with exception of the CCCT circuit, others circuit design, like SVPWM and QEP, are presented in Fig. 11(a) and 11(b), respectively. The SVPWM circuit is designed to be 12 kHz frequency and 1μs dead-band. The circuit of the *QEP* module is shown in Fig.11(b), 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.

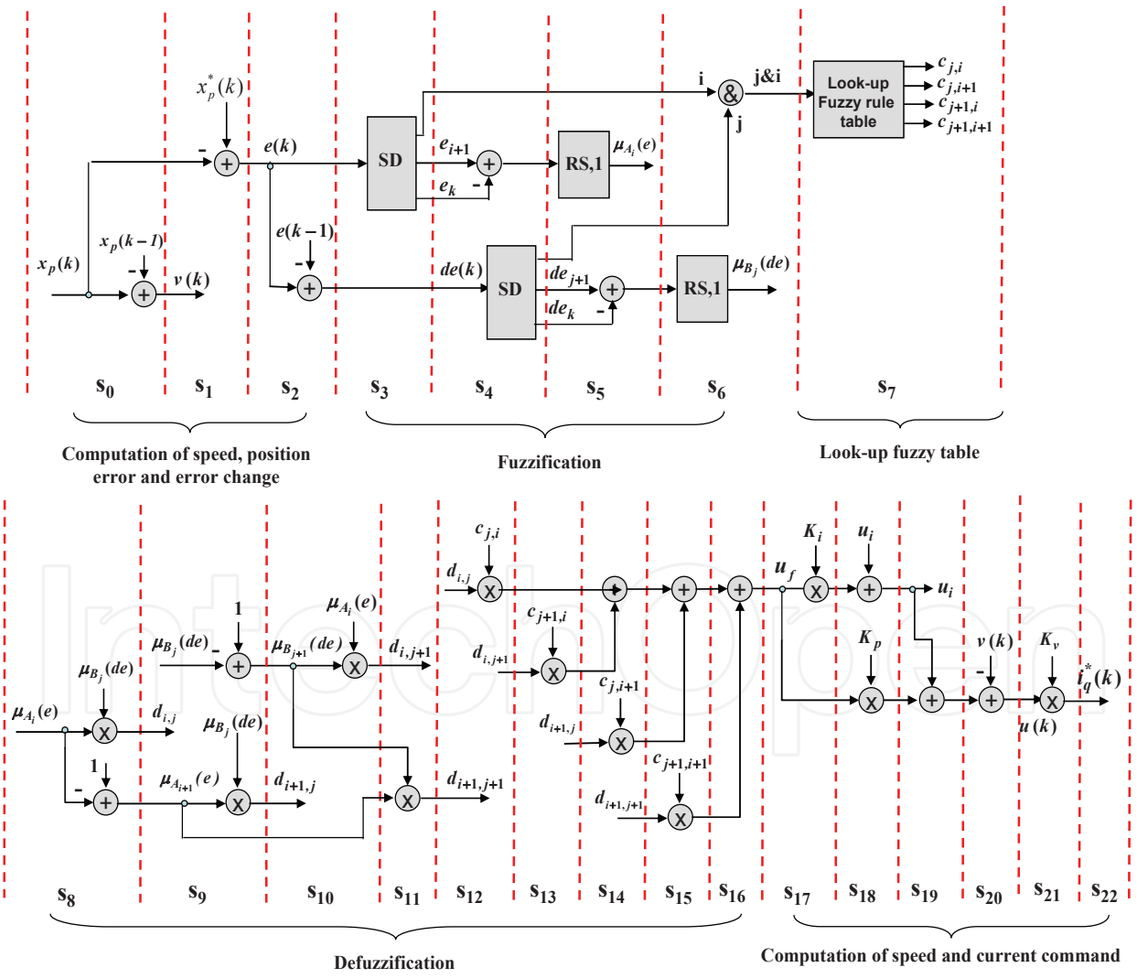


Fig. 9. State diagram of an FSM for describing the FC in position loop and P controller in speed loop

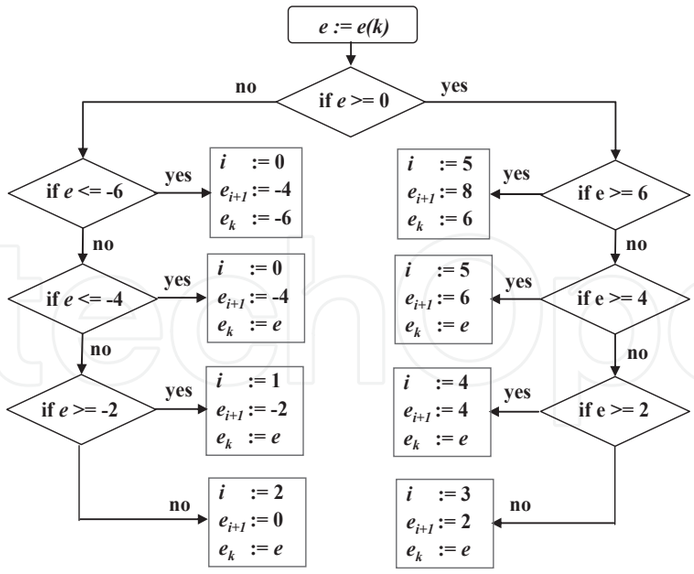


Fig. 10. Section determination in Fig. 9

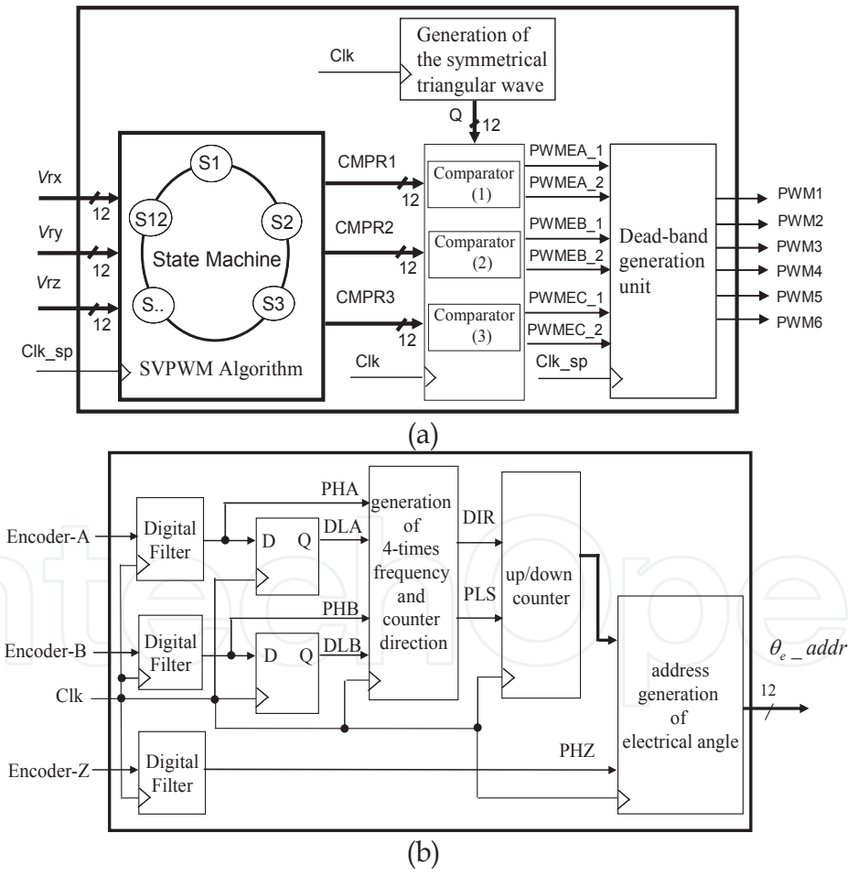


Fig. 11. Block diagram of (a) SVPWM circuit (b) QEP circuit

The Nios II embedded processor IP is depicted to perform the function of the motion trajectory for X-Y table in software. Figure 12 illustrates the flow charts of the main program and the interrupt service routine (ISR), where the interrupt interval is designed with 2ms.

All programs are coded in the C programming language in Fig.10. 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. Using the C language to develop the control algorithm has the portable merit and is easier to transfer the mature code from the other processor to the Nios II embedded processor. Finally, Table 1 shows the FPGA utility of the proposed motion control IC and the overall circuits included a Nios II embedded processor IP (5,059 ALUTs and 78,592 RAM bits) and an application IP (10,196 ALUTs and 102,400 RAM bits), use 31.5% ALUTs resource and 7.1% RAM resource of Stratix II EP2S60.

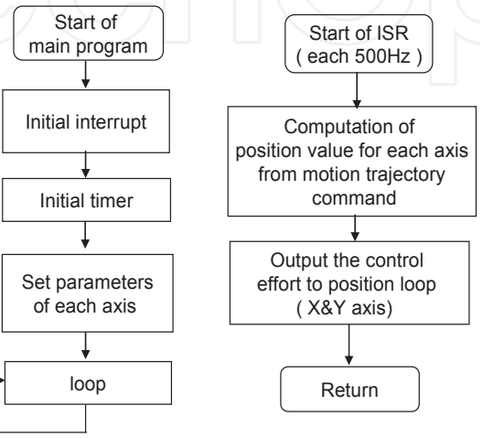


Fig. 12. Flow chart of the main and ISR program in Nios II embedded processor

IP	Module circuit	Logic gate (ALUTs)	Memory (Bits)
Nios II embedded processor IP		5,059	78,592
Application IP (for X-axis and Y-axis)	Position fuzzy controller and speed P controller	1,943 × 2	0
	Current controller and coordinate transformation (CCCT)	649 × 2	49,152 × 2
	SVPWM generation	1,220 × 2	0
	ADC interface	123 × 2	0
	QEP detection and transformation	79 × 4	0
	others	1,005 × 2	2,048 × 2
Total		**15,255**	**180,992**

Table 1. Utility evaluation of a motion contron IC for X-Y table in FPGA

4. Experiments and results

The overall experimental system is depicted in Fig. 1. This system includes an FPGA experimental board, two sets of voltage source IGBT inverter and an X-Y table which is driven by two PMSMs and two ball-screws. The power, rating, voltage, current and rating speed of PMSM are 200W, 92V, 1.6A and 3000rpm, respectively. A 2500 ppr rotary encoder attached to PMSM is used to measure the motor's electrical angle. Two linear encoders with 5μm resolution are mounted on the X-axis and Y-axis table as a position sensor. Each ball-

screw has 5mm lead. The inverter has 6 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 EP2S60 in Fig.1 is used to develop a full digital motion controller for X-Y table. The motion trajectory are implemented by software using Nios II embedded processor, and the two axis position/speed/current vector controller are implemental by hardware in FPGA. In the experimental system, the PWM switching frequency of inverter is designed with 12k Hz, dead-band is 1μs, and the sampling frequency in current loop and position loop of the PMSM are designed with 16kHz and 500Hz, respectively. The motion control algorithms are coded by C language.

In experiment, the position step response and the motion trajectory control are used to evaluate the dynamic performance of the proposed system. In the experiment of the step response, the results of X-axis and Y-axis table under 10 mm amplitude and 0.5Hz square wave command are shown in Fig. 13. The rising time, overshoot and steady-state value in Fig. 13(a) are 110ms, 14% and near 0mm, and in Fig. 13(b) are 90ms, 15% and near 0mm. It reveals that the mass carried in X-axis table is heavier than those in Y-axis table. In the experiment of the motion trajectory tracking, one-dimensional trapezoidal motion trajectory, two-dimensional circular and window motion trajectory are tested and its experimental tracking results are shown in Figs. 14 ~ 16. In one-dimensional motion trajectory, the trapezoidal velocity profile is considered which the acceleration and deceleration is designed with 500mm/s^2, maximum speed is 125mm/s, and the overall displacement is designed with moving from 0 mm to 100 mm position. The trajectory tracking results in each axis corresponding with the aforementioned input commands is shown in Fig. 14. It can be seen that the motion of X-axis and Y-axis table can give a perfect tracking with command target both in position or speed trajectory. Further, in two-dimensional motion trajectory, the circular motion trajectory control with center (60, 60) mm and radius 50mm is evaluated and the tracking errors are the maximum ± 0.55 mm in X-axis, and ± 0.75 mm in Y-axis in Fig. 15. The window motion trajectory designed as Fig.4 and its experimental result is shown in Fig. 16, which also shows the tracking errors maximum ± 0.5 mm in X-axis, and ± 0.9 mm in Y-axis. Therefore, from the experimental results of Figs. 13~16, it demonstrates that the proposed FPGA-based motion controller IC for X-Y table is effective and correct.

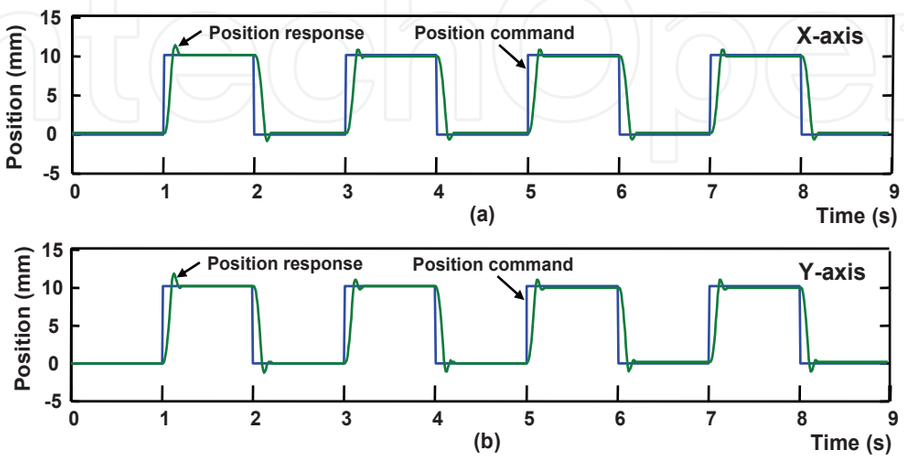


Fig. 13. Step response for (a) X-axis table (b) Y-axis table

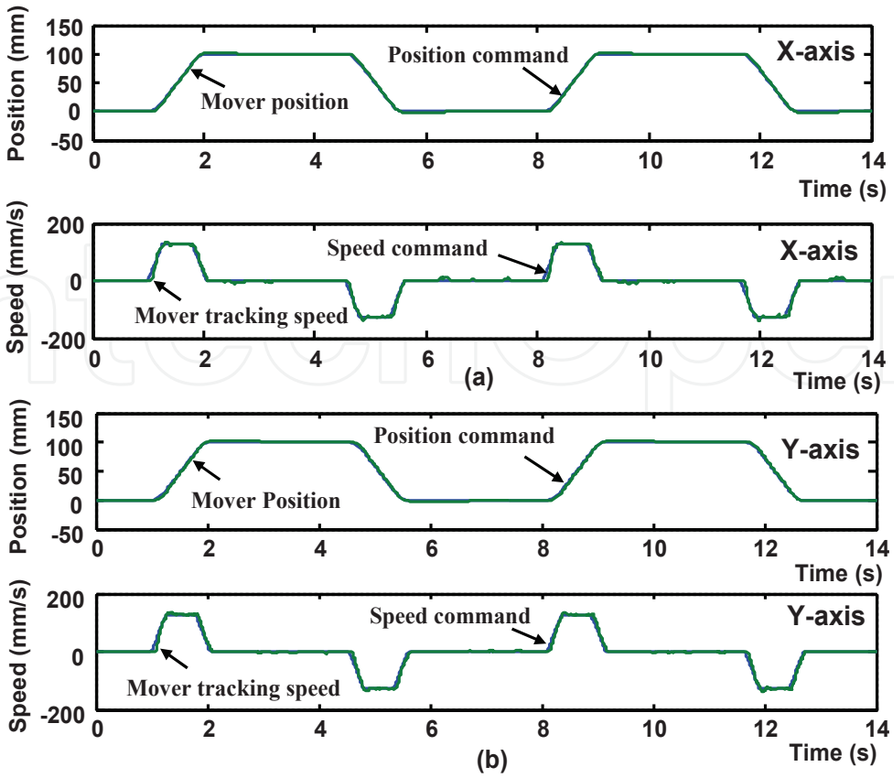


Fig. 14. (a) Position and speed tracking response in X-axis and in (b) Y-axis table

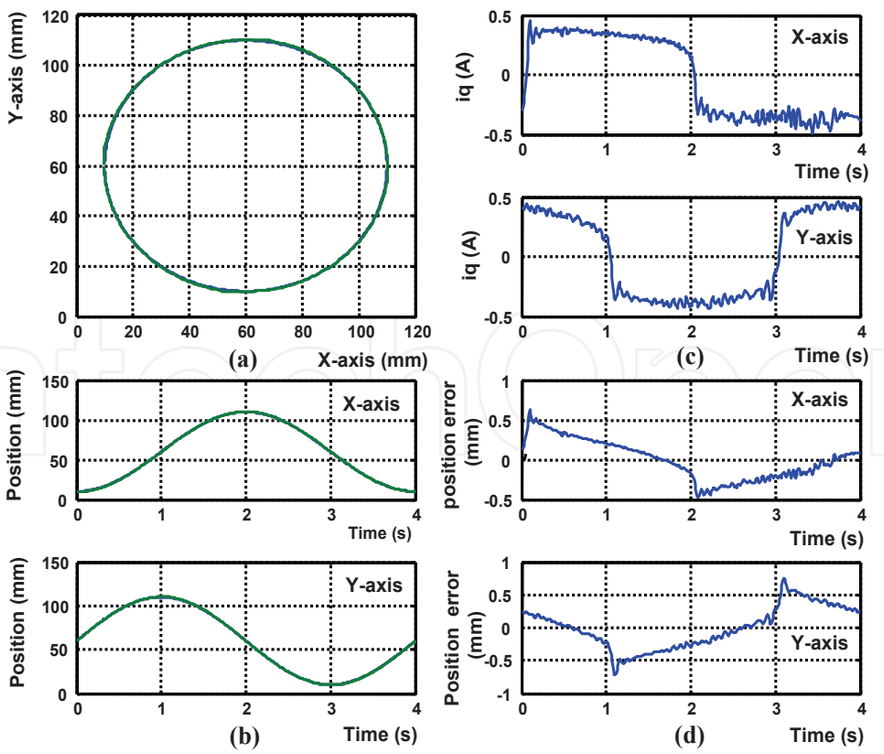


Fig. 15. Response of the circular trajectory (a) circular trajectory response (b) response for X- and Y- axis (c) control effort (d) tracking error

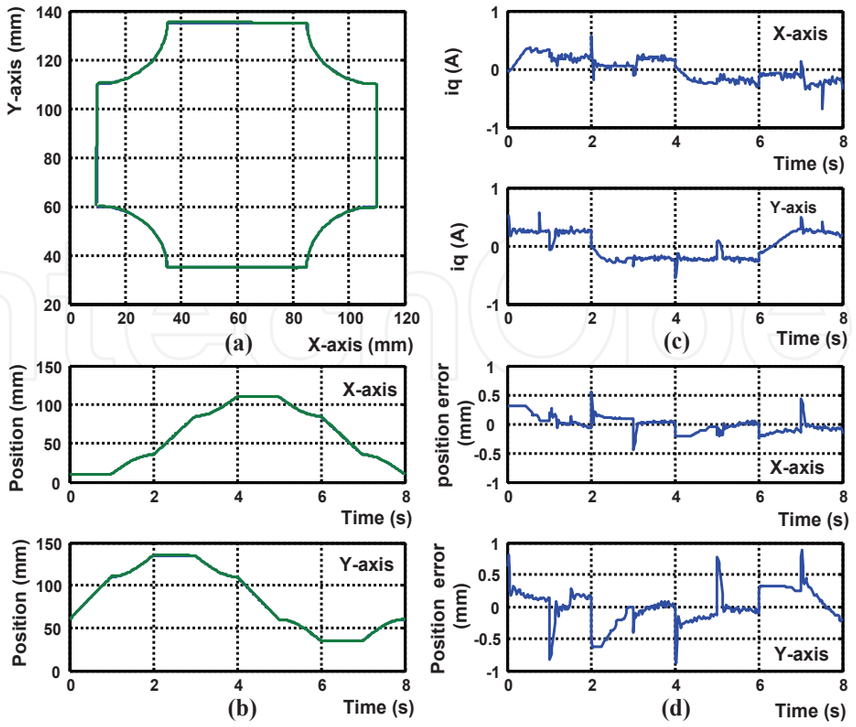


Fig. 16. Response of the window trajectory (a) window trajectory response (b) response for X- and Y- axis (c) control effort (d) tracking error

5. Conclusion

This study successfully presents a motion control IC for X-Y table based on novel FPGA technology. The works herein are summarized as follows.

1. The functionalities required to build a fully digital motion controller of X-Y table, such as the two current vector controllers, two speed P controllers, and two position fuzzy controllers and one motion trajectory planning, have been integrated in one FPGA chip.
2. An FSM joined by one multiplier, one adder, one LUT, or some comparators and registers has been employed to model the overall FC algorithm and the CCCT in vector control of the PMSM, such that it not only is easily implemented by VHDL but also can reduce the FPGA resources usage.
3. The software/hardware co-design technology under SoPC environment has been successfully applied to the motion controller of X-Y table.

However, the experimental results by step response, point-to-point, window and circular motion trajectory tracking, has been revealed that the software/hardware co-design technology with the parallel processing well in the motion control system of X-Y table.

6. References

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

Altera (2008): www.altera.com.

Goto, S.; Nakamura M. & Kyura, N. (1996). Accurate contour control of mechatronic servo systems using gaussian networks, *IEEE Trans. Ind. Electron.*, vol.43, no. 4, pp. 469-476.

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.

Hanafi, D.; Tordon, M. & Katupitiya, J. (2003). An active axis control system for a conventional CNC machine, *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1188-1193.

Hsu, Y.C.; Tsai, K.F.; Liu, J.T. & Lin, E.S. (1996) *VHDL modeling for digital design synthesis*, KLUWER ACADEMIC PUBLISHERS, TOPPAN COMPANY (S) PTE LTD.

Jung, S. & Kim, S.S. (2007). Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems, *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 265-271.

Kung, Y.S.; Huang, P.G. & Chen, C.W. (2004). Development of a SOPC for PMSM drives, *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems*, vol. II, pp. II-329~II-332.

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*, 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.

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. Ind. Electro.*, vol. 50, no.5, pp. 867-880.

Liaw, C.M. & Kung, Y.S. (1999) Fuzzy control with reference model-following response, *Fuzzy Theory Systems: Techniques and Applications*, vol.1, pp. 129-158.

Lin, F.J.; Wang, D.H. & Huang, P.K. (2005). FPGA-based fuzzy sliding mode control for a linear induction motor drive, *IEE Proc.- Electr. Power Application*, vol. 152, no.5, pp. 1137-1148.

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

Naouar, M.W.; Monmasson, E.; Naassani, A.A.; Slama-Belkhodja, I. & Patin, N. (2007). FPGA-based current controllers for AC machine drives – a review, *IEEE Trans. Ind. Electron.*, vol. 54, no.4, pp.1907-1925.

Sanchez-Solano, S.; Cabrera, A.J.; Baturone, I.; Moreno-Velo, F.J. & Brox, M. (2007). FPGA implementation of embedded fuzzy controllers for robotic applications, *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1937-1945.

Wang, G.J. & Lee, T.J. (1999). Neural-network cross-coupled control system with application on circular tracking of linear motor X-Y table, *International Joint Conference on Neural Networks*, pp. 2194-2199.

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 the IEEE/RSJ International Conference on Intelligent Robots and System*, pp. 523~528.

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.

IntechOpen