An FPGA-Based Multiple-Axis Velocity Controller and Stepping Motors Drives Design

Chiu-Keng Lai[1,*], *Wei-Nan* Chien[1], *and Yaw-Ting* Tsao[1]

[1]Department of Electrical Engineering, National Chin-Yi University of Technology
No. 57, Sec. 2, Zhongshan Rd, Taiping Dist., Taichung 41170, Taiwan

Abstract. A Field Programmable Gate Array based system is a great hardware platform to support the implementation of hardware controllers such as PID controller and fuzzy controller. It is also programmed as hardware accelerator to speed up the mathematic calculation and greatly enhance the performance as applied to motor drive and motion control. Furthermore, the open structure of FPGA-based system is suitable for those designs with the ability of parallel processing or soft code processor embedded. In this paper, we apply the FPGA to a multi-axis velocity controller design. The developed system integrated three functions inside the FPGA chip, which are respectively the stepping motor drive, the multi-axis motion controller and the motion planning. Furthermore, an embedded controller with a soft code processor compatible to 8051 micro-control unit (MCU) is built to handle the data transfer between the FPGA board and host PC. The MCU is also used to initialize the motion control and run the interpolator. The designed system is practically applied to a XYZ motion platform which is driven by stepping motors to verify its performance.

1 Introduction

A Field Programmable Gate Array (FPGA) based system is a great hardware platform to support the implementation of controllers such as PID controller, fuzzy controller, adaptive controller, optimum controller, FIR filter and even neuro network system [1-4]. It could also be programmed as an ASIC chip, which combines many functions into one FPGA chip, such as PID controller, velocity profile generation, interpolator, and data conversion [5]. It is also used to speed up the mathematic calculation and greatly enhance the performance of the motion planning and drive design [6-8], or used on the other relative field where hardware calculation is important. In addition, due to the properties of open structure, parallel processing design with FPGA is possible and easily designed [7, 9]. Besides, some IPs of soft code processor for system on a chip (SOC) design are provided, such as the MCU NIOS II, Arduino, or ARM DSP [2-4, 10-13], which could be adopted to easily realize the design of embedded system, or design in co-processor operation. With those IPs, the users could easily program their peripheral circuits connecting to the embedded soft code processor to reach their desired functions and performances.

In this paper, we investigate the feasibility of a motion planning and motion control implemented by FPGA. The proposed motion control chip is based on a soft code processor 8051 and as an embedded system to realize the multi-axis motion control where the velocity control is default to be constrained by the trapezoidal velocity profile. The designed system also realizes the interpolator for multi-axis synchronous operation by adopting the Digital Differential Analyzer (DDA) algorithm [13-16]. Furthermore, the designed hardware system is connected to a host PC through the universal asynchronous receiver transmitter (UART) interface to receive the pulse commands dominated by the human machine interface (HMI) on PC. Such a soft code MCU-based embedded system has several advantages. Since it is a dedicated and open structure platform, it can be optimized to run the motion planning algorithms in the most efficient way. It may also be attached to the host computer to reduce the computation time by sharing the PC resources. The cost of such hardware system or processor would be a fraction of a general purpose CPU, hence hundreds of soft code MCU can be deployed in parallel to further increase the computational capabilities as compared with those controlled by PC only, or have the competitiveness as compared to such systems of PC plus axis motion control cards due to their flexibility in design. Most important of all, by implementing such a processor and system on reconfigurable hardware, and utilizing high level abstraction of the motion planning algorithms, the motion system is easily realized and totally integrated into the embedded system.

The paper is organized in the following manner. In Section 2, the architecture of the proposed system is presented. After that, the designed modules for motion control and velocity profile implementation are shown in Section 3. In Section 4, the stepping motor drive design used to the experiment is shown. In Section 5, the

* Corresponding author: chiukl@ncut.edu.tw

experimental results followed the proposed algorithms are shown by two-axis synchronous operation. Finally, the conclusions are given in Section 6.

2 The Architecture of the Integrated System

The block diagram of the developed system is shown in Fig. 1. It includes a Human Machine Interface (HMI) on PC, the motion controller executed by embedded 8051 MCU, the interpolator, the stepping motor drive system. The developed control system could be applied to three-axis motion system driven by stepping motor, it could also be used to the servo systems which are set in position mode and driven by the A/B phases or pulse/direction commands.

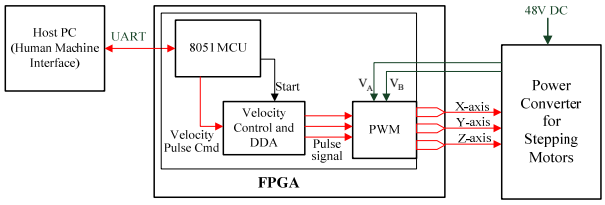


Fig. 1. The architecture of multi-axis motion control system.

The displacement commands for each of the motors are from the HMI on the host PC, and are transferred to the motion controller through the UART interface to generate the stepping pulses under the considerations of trapezoidal velocity profile and multi-axis synchronous operation. After that, the pulse commands are sent to the DDA modules to equally output the stepping pulses to the stepping motor drive system. Finally, the duty cycle controls are added to the stepping commands by passing them through the Pulse Width Modulation (PWM) modules with the current feedbacks, V_A and V_B [17], the motor currents are then regulated on the desired level. As stated above, the overall system integrates the motion planning, velocity control and the stepping motor drive in one chip, which has largely reduced the required complicated structure and the area of PCB.

3 The Motion Control Module Design using FPGA

3.1 The Velocity Programming Module

This module generates the desired acceleration and deceleration performances, and makes the run speed to the permitted maximum feed rate ($V_{\max}$ in pps). The acceleration and deceleration are constrained by the trapezoidal velocity profile as shown in Fig. 2. The sampling time for the testing motion platform is set as $T = 6.5$ ms. Thus, according to the profile shown in Fig. 2, the system maximum feed rate is limited to 180 rpm for X, Y and Z axes. In Fig. 2, the parameters of a is the acceleration and T is the sampling time. To let the system have the same displacements in the acceleration and deceleration durations as Fig. 2 shows, the motors are ideally accelerated to $V_{\max}$ from standstill within duration time 4T, and the duration time is also suitable for the deceleration process. The procedures to generate the velocity command with the trapezoidal velocity profile are as in the following three steps:

(a) In the acceleration duration, the motor is accelerated from the initial speed V_0 with constant acceleration until the motor reaches to the maximum feed rate $V_{\max}$. The amount of pulse in each sampling time is increased by one, which leads to a constant acceleration.

(b) In the constant speed duration, a constant amount of pulse in each sampling time period is generated. Furthermore, the total displacement (in pulse number) and the remaining displacement off the destination are used to decide the time instant for system switching to the deceleration duration. The decisions above-stated are executed by 8051 MCU in FPGA.

(c) In the deceleration duration, the motor speed is decelerated from the $V_{\max}$ to V_0 with constant deceleration rate. In here, deceleration procedure decreases one pulse for each of the sampling time. However, the remaining distance is another consideration when in deceleration duration, and is discussed in subsection 3.3.

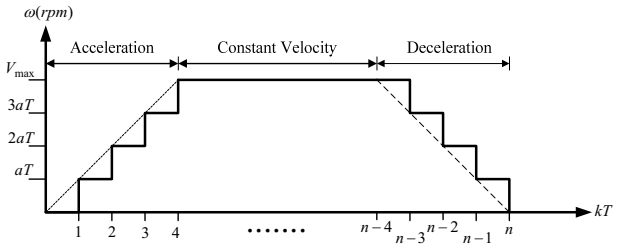


Fig. 2. The trapezoidal velocity profile.

3.2 The Interpolator by Hardware DDA

The parallel processing capability of FPGA makes the multi-axis operation realizable [9] within one chip system. Also, for each of the drive system, its interpolator in here is accomplished by the DDA algorithm. The hardware DDA for one axis, which is shown in Fig. 3, is mainly made by two 8-bit registers and one 8-bit adder. The motion control system accesses the pulse commands processed in last sub-section, and equally sends the stepping pulse to the motor drive system controlled by DDA. In Fig. 3, the output port $\Delta Z[n] = 1$ stands for one stepping signal is occurred.

3.3 The Two-axis Velocity Command Generation

In Fig. 4, it demonstrates the algorithm for two axes, X-axis and Y-axis, motion synchronously, and assumes that the displacement of X-axis (long axis) is greater than Y-axis (short axis). The algorithms are described as follows.

3.3.1 The X-axis Motion Control

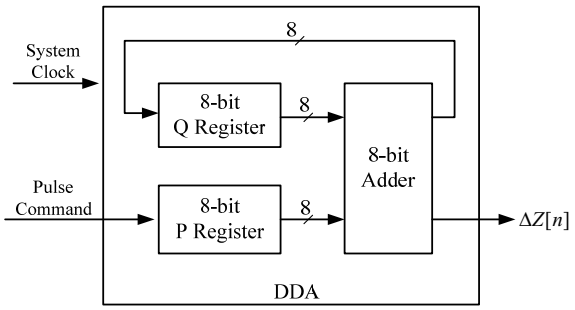


Fig. 3. The block diagram of DDA module.

There are five steps given to demonstrate the generation of stepping pulse for long axis.

Step 1. The pulse commands are stored in the host PC, and the displacements for the two axes are loaded ahead from the host PC to 8051 through the UART interface.

Step 2. The motor dynamics and feed rate are determined by the following equations represented in continuous system,

$$v_X = V_0 + at \tag{1}$$

$$S_X = V_0 t + \frac{1}{2} a t^2 \tag{2}$$

$$v_X^2 = {V_0}^2 + 2aS_X \tag{3}$$

where v_X is the X-axis speed for each of the durations, a is the corresponding acceleration, S_X is the displacement in X-axis and t is the duration time. Motion control by digital controller is usually represented in discrete type. Thus, all the variables shown in Equations (1)~(3) could be represented in discrete type, such as

$$v_X[n] = V_0 + na[n] \tag{4}$$

where $n = 0, 1, 2, \cdots$.

According to the motion control by trapezoidal velocity profile and the dynamics of Equations (1) ~ (3), the motor speed at the three durations shown in Fig. 2 can be decided as follows:

(a) Acceleration duration:

Drive system is accelerated under constant acceleration and the drive speed is represented as

$$v_a[n] = V_0 + na[n] \tag{5}$$

(b) Run duration (Constant velocity):

If the point-to-point displacement is long enough, then the system in run duration may existent, and maximum feed rate is asserted.

$$v_c = V_{\max} \tag{6}$$

(c) Deceleration duration:

Deceleration is made by decreasing the amount of the generated pulse monotonously.

$$v_d[n] = V_{\max} - na[n] \tag{7}$$

In Equations (5)~(7), v_a, v_c and v_d are the velocities for the three durations, and are represented as pulse per sampling duration.

Assuming the displacements of acceleration and deceleration durations are the same, and the constant velocity duration exists, then the total displacements, S_X, could be represented as

$$\begin{aligned} S_X &= S_a + S_c + S_d \\ &\overset{\Delta}{=} L_{des\_X} \end{aligned} \tag{8}$$

where S_a, S_c and S_d are respectively the displacement of acceleration duration, constant velocity duration and deceleration duration, and L_{des_X} is the desired displacement of X-axis for each of the command line. If the desired displacement is long enough, then $V_{\max}$ can be the permitted maximum speed. On the other hand, for a short displacement, $(S_a + S_d) >> S_c$, the motion with trapezoidal velocity profile is modified as triangle velocity profile as shown in Fig. 5. From Eq. (3), the maximum feed speed $V_{\max}$ is thus set as

$$V_{\max} = \sqrt{aL_{des\_X}} \tag{9}$$

Since the displacements have been converted into the amount of pulse, and the stepping motor drive systems are set on position mode, the displacement must be represented in integer. We thus have the maximum velocity in integer as follows:

$$V_{\max} = \left\lfloor \sqrt{aL_{des\_X}} \right\rfloor \tag{10}$$

Step 3. According to Equations (5) and (9), the speed in acceleration duration can be realized and calculated in advance. At the duration, the controller monitors the motor speed, and checks whether the speed is greater than the $V_{\max}$ or not. Once it is true, the motion system is switched to the next duration, constant velocity or deceleration.

Step 4. In the constant velocity duration, the motor runs with the constant speed v_c, and simultaneously 8051 MCU checks the total displacement whether the remaining displacements are smaller than the displacement in the deceleration duration or not, i.e.,

$$\mathrm{Rem}_X = L_{des\_X} - S_a - S_c \le S_d \tag{11}$$

where Rem_X is the remaining displacement to the destination in X-axis. Then, the system goes into deceleration duration.

Step 5. Applying Eq. (7) to implement the deceleration control until the total displacement is reached.

3.3.2 The Y-axis Interpolation

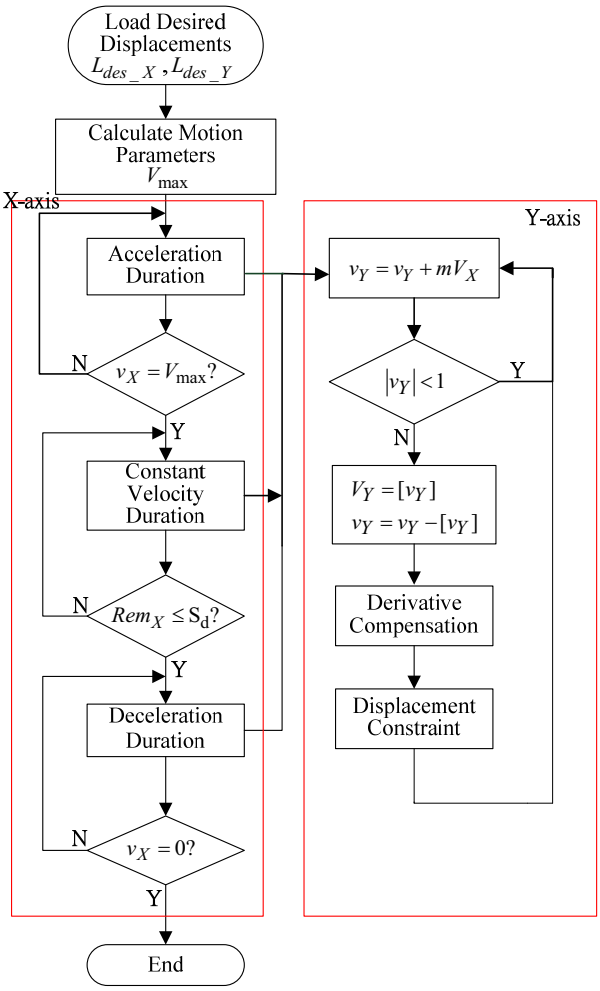


Fig. 4. The algorithm for the two-axis synchronous operation.

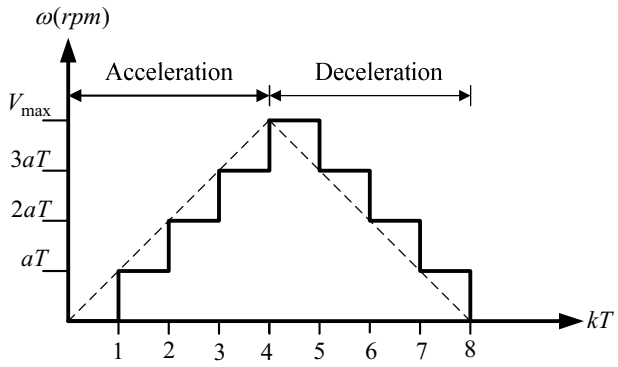


Fig. 5. The triangle velocity profile.

The implementation of Y-axis interpolation is according to the required total displacements, S_Y , and is defined as

$$S_Y = \sum v_Y[n] \tag{12}$$

where $v_Y[n]$ is the speed of Y-axis in the amount of stepping at the sampling instant nT . Define the slope m as

$$m = \frac{L_{des\_Y}}{L_{des\_X}} \tag{13}$$

where L_{des_Y} is the desired Y-axis total displacement. Eq. (13) is used to determine the desired fraction speed of Y-axis as comparing to X-axis. The flow chart is shown in Fig. 4. Since the velocity and position are represented as the amount of pulse, and the remaining displacements are subtracted from the desired position in pps. To avoid the condition of over speed, the algorithm is modified as follow.

Since the motion system is realized by hardware circuit, and finite precision of the data width may cause undesired situations such that the error of displacement may exist. Thus, when we implement the motion control, constraint or compensation is necessary [15]. In the designed system, two conditions may exist during the motion command generation and interpolation. The following is used to compensate those drawbacks.

One of the examples with over speed change on the short-axis (Y-axis) is shown in Fig. 6, where the maximum speed is set to 3 pulses per sampling time. Fig. 6 shows a sudden speed change on Y-axis between the sampling instants 3 and 4. A big sudden speed change may result to the missing of the stepping command owing to the command exceeds the permitted maximum acceleration or deceleration rate.

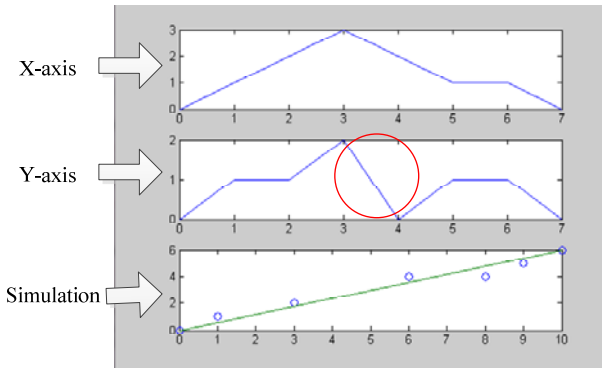


Fig. 6. The simulation results by Matlab without compensation.

To avoid the situation of suddenly large speed change, we add the derivative compensation as shown in Eqs. (14) and (15).

If $\frac{dv_Y}{dt} > a$, then

$$V_Y[n+1] = V_Y[n] + an \tag{14}$$

else

$$V_Y[n+1] = V_Y[n] + (\frac{dv_Y}{dt})n \tag{15}$$

Eq. (14) shows that if the steps 3, 4 and 5 in Fig. 6 lead to the derivative of speed be greater than the permitted maximum acceleration a, the next speed is decided by Eq. (14). Otherwise, the next speed is as the

one of Eq. (15). The simulated example shown in Fig. 6 is modified according to Eqs. (14) and (15), and the results are shown in Fig. 7, where the over speed change is avoided to prevent from missing stepping command.

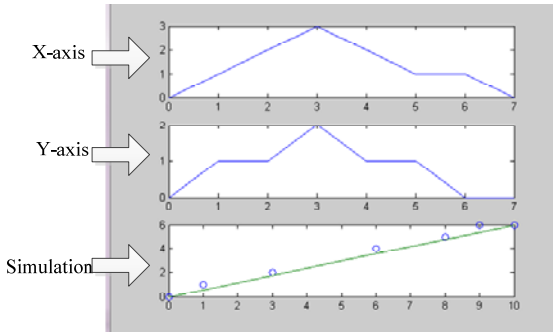


Fig. 7. The simulation results by matlab with acceleration compensation.

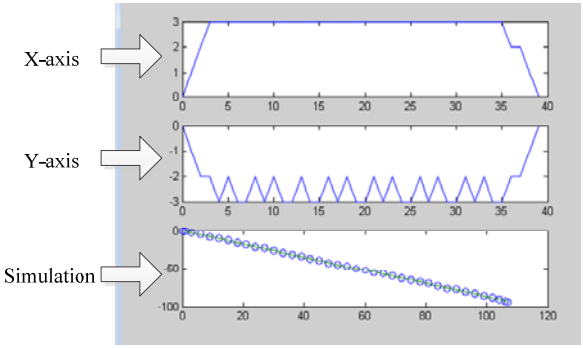


Fig. 8. The simulated results with displacements 107 and -93 on X-axis and Y-axis respectively. It is over interpolation on Y-axis with total displacement -94.

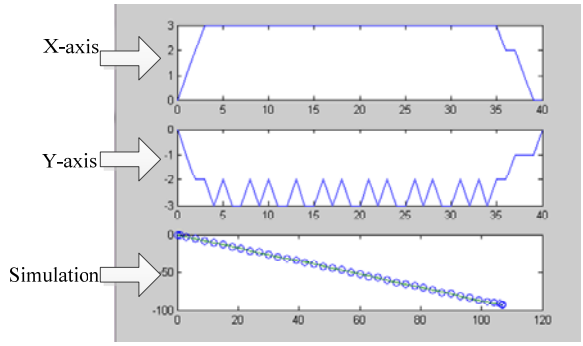


Fig. 9. The simulated results with compensation for the displacements of 107 and -93 on X-axis and Y-axis respectively.

Although the over speed interpolation is reduced with the derivative compensation, the over interpolation may also exist as the simulated results shown in Fig. 8, where the displacement commands on X- and Y-axis are respectively the step numbers of 107 and -93. Thus, if we just consider the over speed change, it is not enough to obtain an exact displacement because of the occurrence of over interpolation.

To avoid the situation of over pulse interpolation, we add the constraints of displacement on the Y-axis as follows:

If $\sqrt{\mathrm{Rem}_Y} > V_Y[n]$, then

$$v_Y = V_Y[n] \tag{16}$$

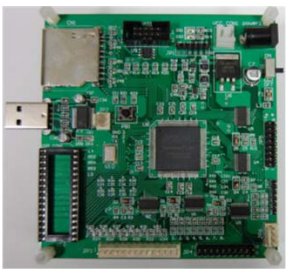
else

$$v_Y = [\sqrt{\mathrm{Rem}_Y}] \tag{17}$$

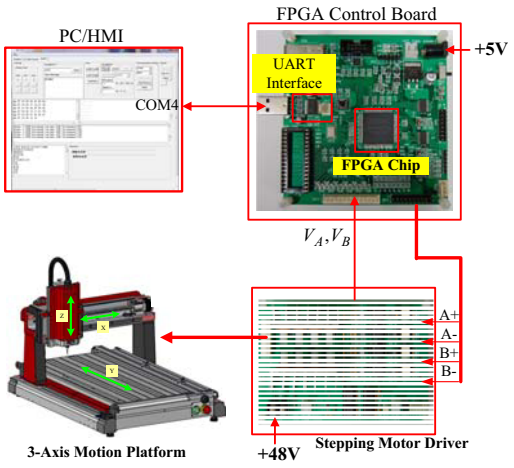
where Rem_Y is the remaining distance in the amount of pulse to the desired position, V_Y is the speed after the derivative compensation, and v_Y is the true interpolation speed. One of the algorithm used to compensate the problem is shown in Fig. 9 where the over interpolator is avoided since the decision of interpolation step is substituted by the remaining displacement of Y-axis.

4 The FPGA-based Control Board and Stepping Motor Drive based Motion Platform

Fig. 10(a) is the developed FPGA control board, which is mainly designed by Altera cyclone III FPGA, and an USB connector is used to communicate with host PC by UART protocol to download the processed G-code commands to the soft code processor 8051 on control board. The baud rate is 19200 bps. In addition, the control system outputs the stepping signals from the FPGA interface to the stepping motor drive system to drive the motors. The overall system setup is shown in Fig. 10(b).



(a)



(b)

Fig. 10. (a) Control board, and (b) experimental setup.

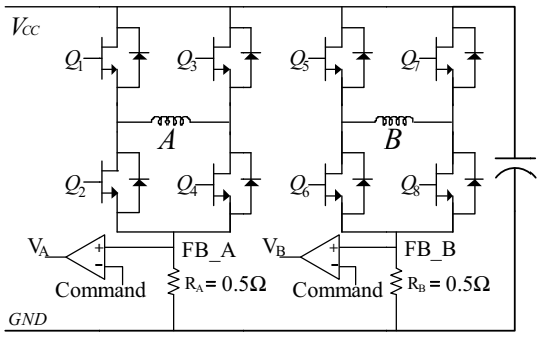


Fig. 11. The power converter of hybrid stepping motor.

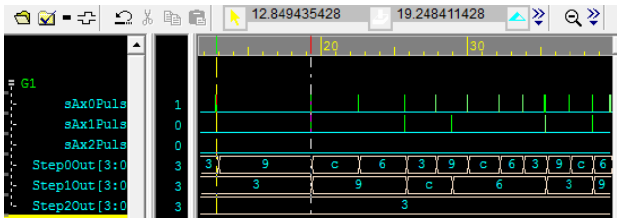


Fig. 12. Stepping signals for stepping motor drive.

The circuit of power converter for hybrid stepping motor drive is shown in Fig. 11. The DC bus voltage is 48 V and current command is set as 1 A. The stepping motors are operated in full-step mode and winding current detection is executed by the two resistors, R_A and R_B, as shown in Fig. 11. The winding current is compared by linear comparators, and their outputs (V_A and V_B) are used to decide the duty cycle of stepping motor drives. The PWM modules are operated in 20 kHz sampling frequency. The detail operations could be found in [17].

5 Simulations and Experimental Results

We apply those considerations and hardware setup stated above to the experiment of two-axis control. They are run under the conditions of six straight lines with different slops to evaluate whether the algorithms set well or not. The six lines are with the following coordinates: {(0,0), (5,5)}, {(1,0), (5,5)}, {(2,0), (5,5)}, {(3,0), (5,5)}; {(4,0), (5,5)}; {(5,0), (5,5)}. The stepping signals for stepping motors are simulated in Fig. 12, where two-axis operation with DDA is shown. The DDA output ports are sAx0Puls and sAx1Puls, and the stepping signals Step0Out[3:0] and Step1Out[3:0] are the signals for the hybrid stepping motor in full step moving. Note that in Fig. 12, the stepping signal for axis 3 isn't activated. The source G-codes are simulated as shown in Fig. 13.

Regarding to the practical experiment by the caving machine as the motion control platform, the system is designed to send the velocity commands from the host PC to the motion controller on the FPGA board through the UART interface. Two conditions are demonstrated by adopting the DDA and without it. Fig. 14 is the one which just executes the trapezoidal velocity motion control and works at two-axis operation, but the DDA modules isn't included. While in Fig. 15, the DDA modules are respectively added to X-axis and Y-axis. The experimental results show that the six lines are straight than those without the DDA. The motion control, interpolation and stepping motor drive are integrated and work well. The experimental results are as expected.

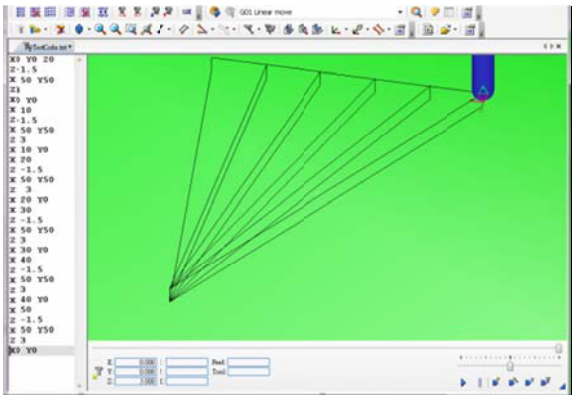


Fig. 13. The simulated results.

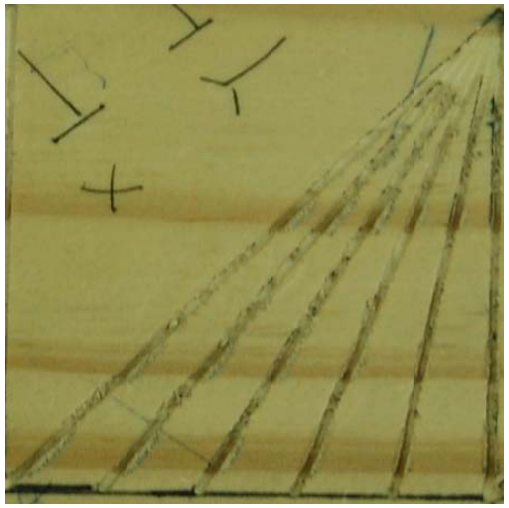


Fig. 14. The experimental result without DDA.

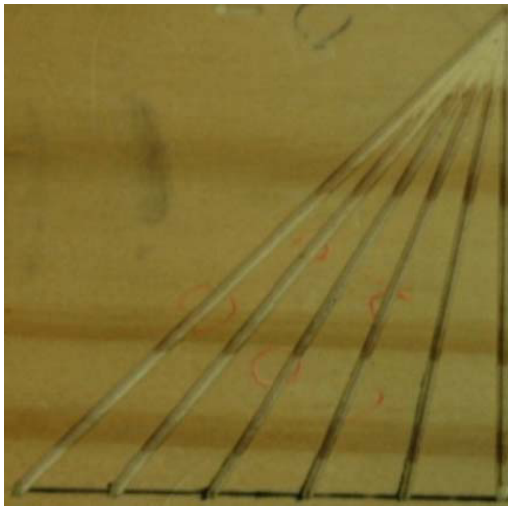


Fig. 15. The experimental result with DDA.

6 Conclusions

In the paper, we design the multi-axis velocity control system using the FPGA. The system includes a soft code processor 8051, the UART interface, the motion planning and interpolation, and the stepping motor drive. Simulations and experiments are demonstrated on a two-axis operation. The simulation and experimental results show that the developed multi-axis motion control system has reached the desired performance about the motion control and two axes operation synchronously. To improve the processing performance, a fast baud rate of communication is necessary. We are going to add the USB communication protocol to the system to enhance the performance of data transfer. As regarding to the embedded MCU 8051, it has the most instructions been run in one clock machine cycle. For further requirement on the processing performance, another MCU such as ARM could be adopted to have a better the performance.

References

1. N.K. Quang, Y.S. Kung, Q.P. Ha, “FPGA-based control architecture Integration for Multi-axis Tracking Motion System”, IEEE/SICE 2011, pp. 591-596 (2011).
2. R. Nithya, K.R. Sarath Chandran, V.P. Chandramani, “Run-Time Reconfiguration of Processing Elements Through Soft-Core Processor”, 2014 International Conference on Communications and Signal Processing (ICCSP 2014), pp. 813-817 (2014).
3. H.P. Sekaran, M.H. Liyanage, N. Krouglicof, “Optimization of Controller Gains for FPGA-based Multi Variable Motion Controller using Response Surface Methodology”, IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE 2015), pp. 1307-1312 (2015).
4. Z. Szadkowski, E.D. Fraenkel, A.M. Van Den Berg, “FPGA/NIOS Implementation of an Adaptive FIR Filter using Linear Prediction to Reduce Narrow Band RFI for Radio Detection of Cosmic Rays”, IEEE Transactions on Nuclear Science, 60(5), pp. 3483-3490 (2013).
5. J.U. Cho, Q.N. Le, J.W. Jeon, “A FPGA-based Multiple-Axis Motion Control Chip”, IEEE/IE, 56(3), pp. 856-870 (2009).
6. N. Atay, B. Bayazit, “A Motion Planning Processor on Reconfigurable Hardware”, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, pp. 125-132 (2006).
7. A. Ghasemazar, M. Goli, A. Afzali-Kusha, “Embedded Complex Floating Point Hardware Acceleration”, 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems, pp. 318-323 (2014).
8. W. Wang, K. Guo, M. Gu, Y. Ma, Y. Wang, “A Universal FPGA-based Floating-point Matrix Processor for Mobile Systems”, 2014 International Conference on Field-Programmable Technoloy (FPT2014), pp. 139-146 (2014).
9. C.K. Lai, K.L. Ho, “To Develop a Parallel Processing System with the FPGA for Multi-Axis Motion Control Platform”, Applied Mechanics and Materials, 284-287, pp. 2396-2401 (2012).
10. J. Zhang, X. Wang, “Position Loop Method in Multi-axis Motion Controller using Extended DDA Circuit”, IEEE International Conference on Automation and Logistics (ICAL '09), pp. 692-696 (2009).
11. X. Yang, Y. Song, P. Wang, “Research and Implementation of a Portable Character Recognition Device”, International Conference on Computational and Information Sciences (ICCIS 2013), pp. 1815-1818 (2013).
12. H. Sun, Y. Zhang, J.J. Xue, Z. Wu, “The Remote Control System of the Manipulator”, 33rd Chinese Control Conference (CCC 2014), pp. 8283-8286 (2014).
13. P. Desai Dev, “Designing and Improving the Efficiency of Hardware Interpolator”, International Conference on Computing, Communication and Automation (ICCCA 2015), pp. 959-962 (2015).
14. C.Y. Chen, P.S. Liao, C.C. Cheng, G.F. Jong, “Design and implementation of integrated non-uniform rational B-spline and digital differential analyzer interpolators for computerized numerical control servo-controllers”, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 221(9), pp. 1075-1087 (2007).
15. K.H. Su, C.K. Hu, M.Y. Cheng, “Design and Implementation of an FPGA-based Motion Command Generation Chip”, IEEE International Conference on Systems, Man and Cybernetics (SMC '06), 6, pp. 5030-5035 (2006).
16. W.H. Jiang, H. Wang, “Design of Three-axis Servo System Based on Linear Motor”, 2010 International Conference on Computer Design and Applications (ICCDA 2010), 3, pp. V3-182 - V3-185 (2010).
17. C.K. Lai, W.N. Chen; S.L. Tsai, Y.T. Tsao, “Simulation and Hardware Implementation of Stepping Motor Current Regulator by Matlab Simulink and FPGA”, International Conference on Electrical and Electronics (EEE2014), Hong Kong, 25-27 April (2014).