Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 301646, 9 pages
http://dx.doi.org/10.1155/2014/301646

Research Article

An Implement of FPGA Based PCI Controller Device and Improvement of DDA Arc Interpolation

Zhengjie Zhang[1] and Zhandong Yu[2]

[1] *School of Astronautics, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China*
[2] *College of Engineering, Bohai University, Jinzhou, Liaoning 121013, China*

Correspondence should be addressed to Zhengjie Zhang; zengjie.zhang.china@gmail.com

Received 17 April 2014; Accepted 2 June 2014; Published 6 August 2014

Academic Editor: Josep M. Rossell

The main purpose of this paper is to develop a new kind of PCI slave device serving as a motion controller for a biaxial motion control system. This kind of controller device is a new realization scheme of PCI devices, which is embedded with a deeply customized PCI interface block instead of traditional PCI interface chips, which will greatly promote the comprehensive performance of the device. Besides, we improved the popular and widely used DDA arc interpolation algorithm, promoting its performance in both accuracy and stability, and integrated it into our device, allowing the ability of the moving parts to move along nonlinear curve paths. Currently, this kind of controller device has been successfully applied on a surface mount machine which is also developed by our lab. As a result, the controller device performs well and is able to satisfy the requirement of accuracy and velocity of the surface mount machine. And its reliability and stability are also remarkable.

1. Introduction

Biaxial motion control system is a kind of electromechanical system widely used in various areas such as industrial manufacturing and commodity production [1]. For example, plane coordinate plotter and surface mount machine are two representative kinds of biaxial systems. In general, one of the common points of this kind of systems is the requirement of high speed and accuracy.

For example, a plane coordinate plotter mentioned in [2] is a typical kind of biaxial system. The system consists of several components including a motion platform, turn-screws, stepping motors, plotting cursor, microcontroller, and computer interface. During a plotting process, the cursor is moved to a given position under the control of the microcontroller according to the instructions from the computer [2]. In order to draw a wanted curve quickly and accurately, high speed of data transaction must be guaranteed as well as the reliability. Surface mount machine is also a kind of sophisticated biaxial motion system [3], the index requirement of which is even higher.

Here in this paper, an implementation of motion control board specially intended for biaxial motion systems is proposed. This control board is designed as a slave device abiding PCI bus protocol, allowing fast data transactions between the upper control master and the slave device, possibly reaching to a peak speed of 132 MB/s (32-bit in 33 MHz clock).

In terms of board level design, traditional scheme usually includes microcontroller, FPGA, and an extra PCI interface device, such as PLX9054 [4, 5]. However, this kind of design is not compact enough. We embedded the PCI protocol decoding block, written in Verilog HDL, into a FPGA device on board instead of independent PCI interface chips, reducing a large quantity of routes on board, thus promoting the reliability of the board system and cutting down the total cost. More importantly, the board is convenient to update to a new scheme version and can be deeply customized according to the demand of customers. Furthermore, the compact design is easy to be protected from imitation and malicious plagiarism.

Another important issue which ought to be carefully considered for biaxial motion system is arc motion, that is, how to move the cursor along a curve rather than a line [6], which requires a systematic moving algorithm to arrange the velocities and positions of the two motors of the two axes.

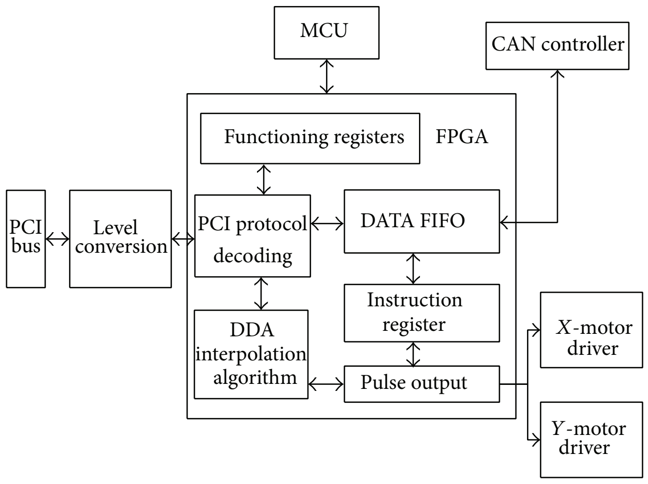

Figure 1: Systematic structure of the control board.

To solve this problem, an algorithm called digital differential analyser (DDA for short) has been proposed [7, 8]. DDA is often used as a motion interpolation algorithm. When moving the cursor of a biaxial motion system, an X-Y plotter, for example, DDA breaks the track into several micro parts. Each of the parts can be covered by one step of the cursor, which is realised by moving the cursor to a micro distance along one axis. As a result, a curve can be covered within several micro steps. The basic principle of DDA arc interpolation will be discussed later.

Although the basic problem of arc motion has been solved by means of DDA arc interpolation algorithm, some other problems such as moving stability and smoothness still affect the performance of the algorithm. For example, this kind of DDA algorithm, which we call traditional DDA, tends to cause unwanted sawteeth on the moving track and often produces huge errors when arc radius is large. In order to overcome its defects and improve its performance, we modified the algorithm, allowing the moving parts of a motion system to track along a given curve within tolerable errors.

The improved DDA algorithm is rewritten in Verilog HDL, and embedded in an FPGA device on board.

2. Device System Structure

The entire system structure of the motion control board is shown in Figure 1. This board communicates with the upper system through PCI system bus, which offers a rather large bandwidth of data transmission [9]. Meanwhile, the board works under the control of the microcontroller unit (MCU for short) and drives the motors with pulse signals produced by the DDA interpolation block or distributed by the upper system.

2.1. FPGA Device. As shown in Figure 1, most of the functioning logic modules are implemented in FPGA device and written in Verilog HDL. By doing this, the out-chip routing is greatly reduced and the reliability of the entire board system is deeply enhanced.

2.1.1. PCI Protocol Decoding Block. PCI is a multiplexing bus, which makes it relatively sophisticated to decode the PCI protocol. In general, the main task of the protocol decoding block is to separate address and data from the multiplexed AD pins [10], which will be explained in detail in the following contents.

2.1.2. DDA Interpolation Block. DDA interpolation block used in this board is written in Verilog HDL and modularized in FPGA device. Thus, the speed of operation is higher and fewer resources are occupied. This part is also going to be amply discussed in the following contents.

2.1.3. Functioning Registers. By writing data to the functioning registers during I/O transactions, the system is able to control the board device in different modes, which makes the board system rather flexible to use.

2.1.4. DATA FIFO. Since transactions on PCI bus are far faster than those on back-end bus [11], a FIFO or RAM block is necessary to serve as a buffer in order to balance the speed difference.

2.1.5. Instruction Register and Pulse Output Register. The instruction register conserves the control instructions delivered from the system and thus produces a series of frequency-controlled pulses according to the instructions to X-axis and Y-axis motor drivers, driving the cursors to move to the designated position.

2.2. Level Conversion Chips. The signal level on PCI bus is 5 V-TTL, while it is 3.3 V-CMOS on pins of FPGA device. Thus, bidirection bus switches such as 74CBT3384 are needed to serve as level converters between the two different levels.

2.3. Microcontroller Unit. Microcontroller unit (MCU) serves as a center controller, making the system function according to the program written inside the chip.

2.4. Other Essential Components. Components on board also include CAN controller and other bus connectors.

3. PCI Bus Protocol Decoding Block

Although some PCI interface chips, such as PCI9054 [12], are available for this system, we use the PCI protocol decoding block embedded in FPGA device as the PCI bus interface.

3.1. Functions of Decoding Block

3.1.1. Device Configuration. PCI protocol decoding block provides necessary information to PCI master system when the system starts and raises itself, including device ID, vendor ID, resource requirement, and function options [13]. After this, the base address of memory resource assigned by system

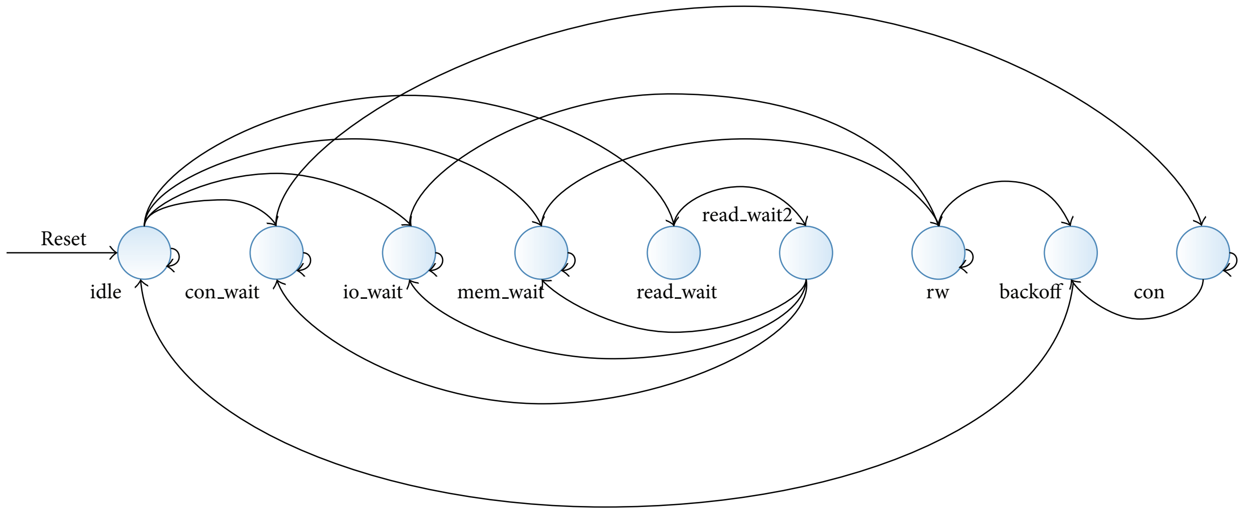

Figure 2: State transition of the PCI protocol decoding block.

is written back into PCI device, which will be conserved by the decoding block.

3.1.2. Address Decoding. When an access occurs on the PCI bus, the PCI device should check out whether it is being called by the system [14]. If the access address on the bus hits the range of the address space of the back-end device, it should respond to the system immediately (usually within 3 cycles according to PCI protocol [15]).

3.1.3. Protocol Timing Decoding. Timing decoding of PCI protocol is the key function of the decoding block. PCI device behaves strictly according to PCI timing.

3.1.4. Bus I/O Control. Protocol decoding block will avoid the occurrence of bus collision by handling enable signals of PCI bus appropriately.

3.2. State Transition. The kernel of PCI protocol decoding block is a state machine. Here are the states involved during a data transaction. And the state transitions of the decoding block are shown in Figure 2.

3.2.1. Idle. The slave device is idle, waiting for an access initiated by the system.

3.2.2. Con_Hold. The system has initiated a configuration access to PCI device, including configuration read and write operations, and is waiting for respond.

3.2.3. I/O_Hold. The system has initiated an I/O access to PCI device, including I/O read and write operations, and is waiting for respond.

3.2.4. Mem_Hold. The system has initiated a memory access to PCI device, including memory read and write operations, and is waiting for response.

3.2.5. Read_Hold. This state is specially inserted between address cycle and the first data cycle in a read access on PCI bus to avoid bus collision.

3.2.6. Configuring. A configuration transmission is taking place.

3.2.7. Rwing. A nonconfiguration transmission is taking place.

3.2.8. Ending. A PCI access is ending. All control signals will be disabled and all S/T/S signals will be released in one cycle.

3.3. Read and Write Operations on PCI. An access on PCI bus consists of three parts, one address cycle, at least one cycle, and several wait cycles. Address appears on AD pins during the address cycle while the data appears during data cycles. And wait cycles are inserted for data latency [16].

There are 3 types of transactions on PCI bus. A configuration transaction usually occurs as soon as the control board is inserted to the system motherboard, while I/O transactions are used for parameter settings. And a memory transaction takes place during a data transition operation.

4. Embedded DDA Interpolation Algorithm

Digital differential analyser (DDA), usually serving as an interpolation algorithm, is widely applied in modern numerical control systems [17]. It is used when shifting the moving parts, or cursors of a system, to a designated position along given tracks, especially curve tracks. Generally speaking, the paths of the cursors controlled by numerical signals will not perfectly match the given continuous track. Therefore, the main issue lies in how to plan a path for a cursor to approach the given track as closely as possible. Based on integral theory, DDA arc interpolation algorithm breaks a continuous track

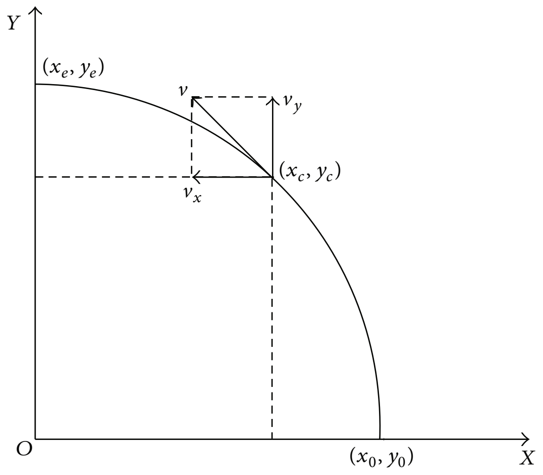

FIGURE 3: DDA arc interpolation in X-Y coordinate.

into a series of discrete points that the cursor of a numerical system is able to reach [18].

4.1. Basic Principle of DDA Arc Interpolation. DDA interpolation algorithm for an arc in Cartesian coordinate is shown in Figure 3 [19]. To make it easy to analyse, we take the 1st quadrant for example. According to Figure 3, we have [19]

$$\frac{v}{R} = \frac{v_x}{y_c} = \frac{v_y}{x_c} = K, \tag{1}$$

where (x_0, y_0) is the starting point and (x_e, y_e) is the ending point, while (x_c, y_c) stands for the current position of the cursor. R is radius of the arc, while v is the tangential velocity. v_x and v_y, respectively, stand for velocity along X-axis and Y-axis. And K is a proportional constant if we assume that the tangential velocity v of the moving part is constant. Therefore, we have [18]

$$\begin{aligned} \Delta x &= v_x \Delta t = K y \Delta t, \\ \Delta y &= v_y \Delta t = K x \Delta t. \end{aligned} \tag{2}$$

Considering this

$$x_e = \int_{(x_0,y_0)}^{(x_e,y_e)} \Delta x\, dx, \qquad y_e = \int_{(x_0,y_0)}^{(x_e,y_e)} \Delta y\, dy, \tag{3}$$

we have [17]

$$x_e = \sum_{i=1}^{m} K y_c \Delta t, \qquad y_e = \sum_{i=1}^{m} K z_c \Delta t, \tag{4}$$

where m is the number of steps it takes for the cursor to reach the ending point (x_e, y_e) starting from (x_0, y_0).

According to formula (4), we get the DDA arc interpolation algorithm [18]. In order to describe the algorithm briefly, we define two arithmetic expressions as follows:

$$A = IF(B), \tag{5}$$

where B is a logic expression, and A equals "1" when B is true and "0" when B is fault. Also, we define

$$A = B \bmod C, \tag{6}$$

where A is the remainder when B is divided by C.

At the beginning, the X-axis integrand register is loaded with y_c, while the Y-axis integrand register is loaded with x_c. At the same time, the accumulators of the two axes are usually half-loaded [20]. In other words, the highest bit is set to "1," while other bits remain "0." Therefore, the initial conditions can be written as

$$\begin{aligned} x_0 &= x_s \qquad & y_0 &= y_s \\ Ax_0 &= 2^{n-1} \qquad & Ay_0 &= 2^{n-1}, \end{aligned} \tag{7}$$

and then the integral clock starts to drive the accumulators to add to the values conserved in the corresponding integrand registers, producing overflow pulses, which drive the integrand registers to update their values with new current coordinate (x_c, y_c). Thus, we have the recursion formulae as

$$\begin{aligned} x_{n+1} &= x_n + \mu_x IF\left(Ax_n + y_n > 2^n\right) \\ y_{n+1} &= y_n + \mu_y IF\left(Ay_n + x_n > 2^n\right) \\ Ax_{n+1} &= \left(Ax_n + y_n\right) \bmod 2^n \\ Ay_{n+1} &= \left(Ay_n + x_n\right) \bmod 2^n. \end{aligned} \tag{8}$$

The algorithm keeps on conducting until the error check registers indicate that the cursor has reached the ending point, or within tolerable errors, after which the iteration stops. Thus, the ending condition can be expressed as

$$\text{End} = IF\left(x_n = x_e,\ y_n = y_e\right). \tag{9}$$

When End equals "1," the recursion stops. And the points (x_n, y_n), forming the path of the cursor, are just what we want.

It is worth mentioning that left-shifting normalizing is often used to maintain velocity stability [21], which will not be deeply discussed here.

Given starting point (8,0), ending point (0,8), and 1/4 arc in the first quadrant, anticlockwise, the simulation result is shown in Figure 4.

As mentioned in Figure 4, we call this kind of DDA algorithm as traditional DDA algorithm. And the logic structure of traditional DDA arc interpolation block embedded in FPGA is shown in Figure 5.

4.2. Improved DDA Arc Interpolation. For the traditional DDA arc interpolation algorithm mentioned in Figure 4, there are some problems. The most fatal one is that when the radius of the arc is far larger than the step length, the errors can be intolerable. For instance, when step length is 1, while radius is 100, the simulation result is shown in Figure 6.

One way to solve this problem is to select an appropriate weighting factor to be multiplied by the integrand before being added into the corresponding accumulator. Here, we

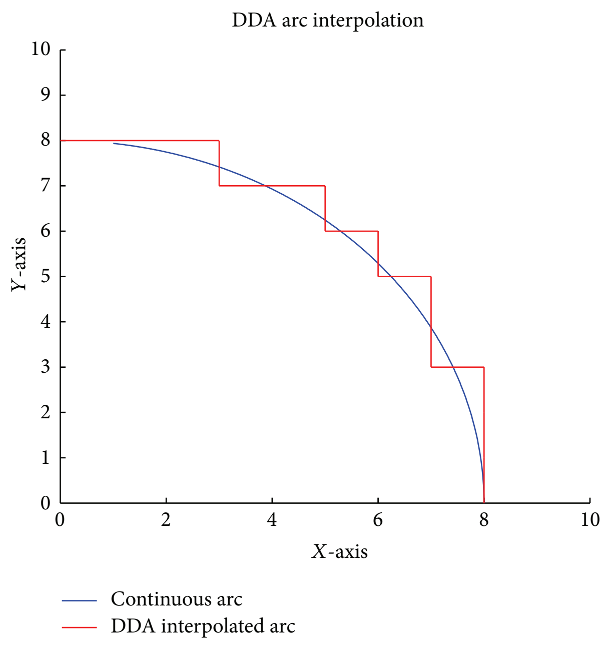

FIGURE 4: Example of DDA arc interpolation.

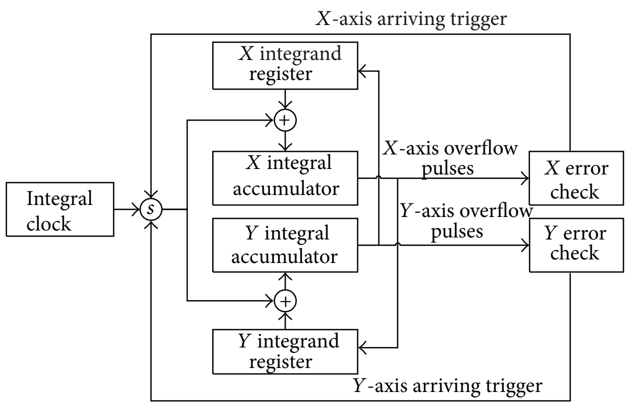

FIGURE 5: Logic structure of traditional DDA arc interpolation.

define the weighting factor as λ. Thus, the recursion formulae (8) can be changed into

$$\begin{aligned} x_{n+1} &= x_n + \mu_x IF\left(Ax_n + \lambda y_n > 2^n\right) \\ y_{n+1} &= y_n + \mu_y IF\left(Ay_n + \lambda x_n > 2^n\right) \\ Ax_{n+1} &= \left(Ax_n + y_n\right) \bmod 2^n \\ Ay_{n+1} &= \left(Ay_n + x_n\right) \bmod 2^n. \end{aligned} \tag{10}$$

When $\lambda = 0.125$, while other conditions are the same as Figure 6, the simulation result is shown in Figure 7, from which it can be concluded that a properly small λ can improve the performance of DDA when radius is large. We call this kind of DDA algorithm as weighted DDA arc interpolation.

Another problem is that even though the errors are small enough, a great number of "sawteeth" can be seen on

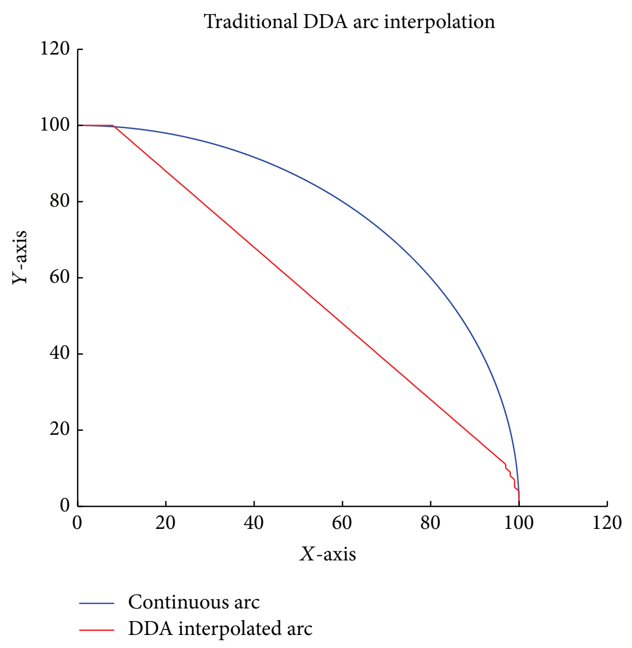

FIGURE 6: Huge errors of traditional DDA algorithm.

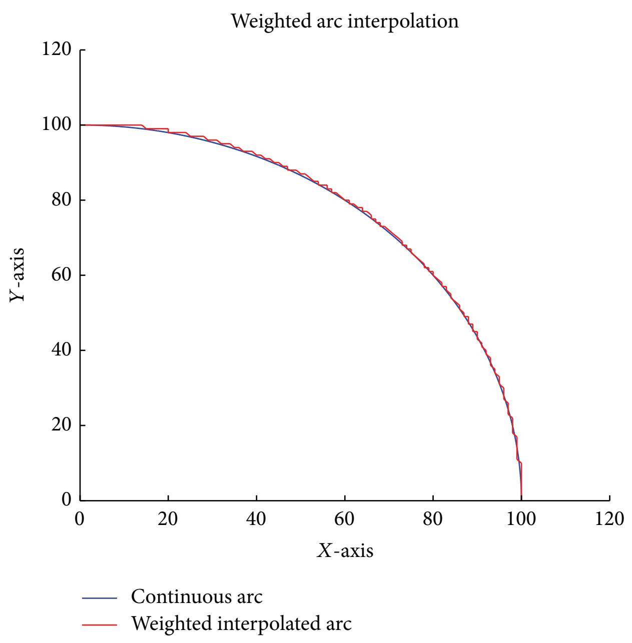

FIGURE 7: Better performance of weighted DDA algorithm.

the path as shown in Figure 8, which may cause constant mechanical shocks on the system. The main reason leading to this problem is that the accumulators of the 2 axes function separately. Thus, the motion on each axis proceeds separately as well, unless both the accumulators overflow at the same time.

Generally, a sawtooth consists of a y-axis step motion and an x-axis step motion closely following as shown in Figure 9. It is obvious that a sawtooth always occurs when accumulator

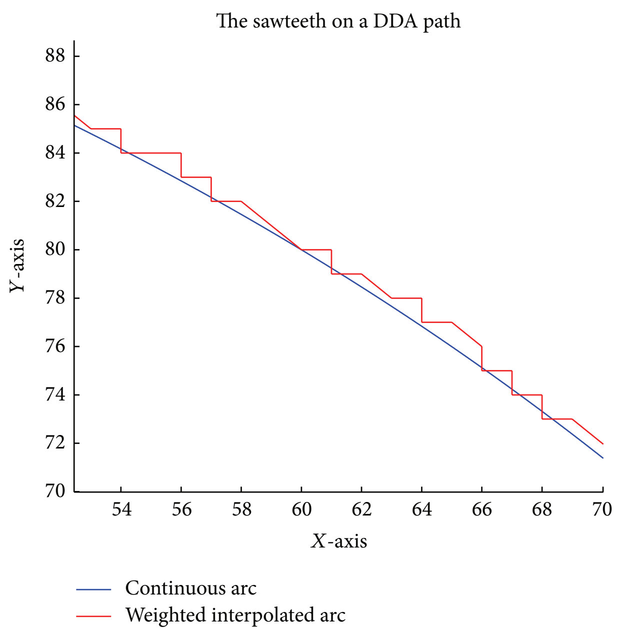

FIGURE 8: The sawteeth of DDA.

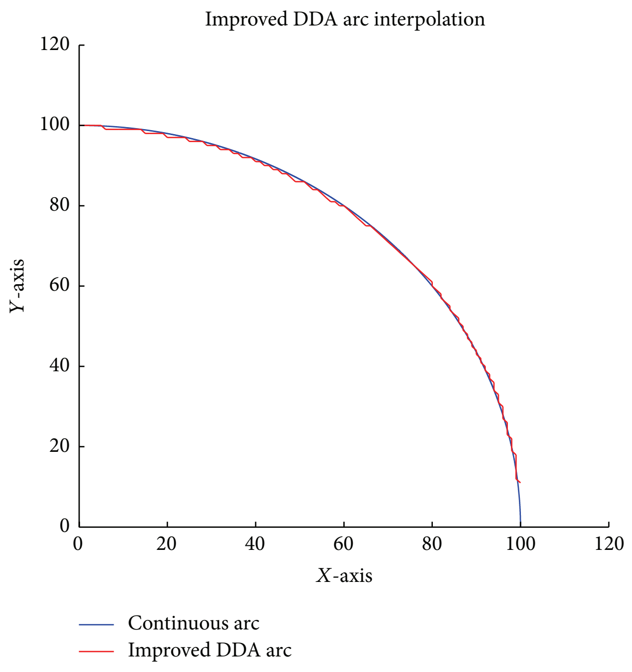

FIGURE 10: Excellent performance of improved DDA algorithm.

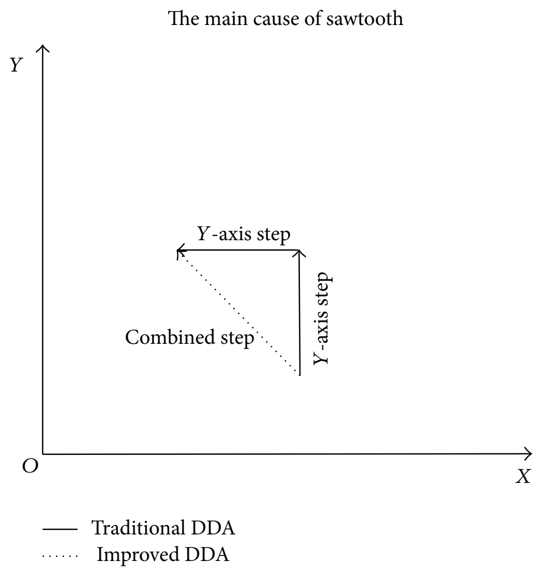

FIGURE 9: Sawtooth along the arc path.

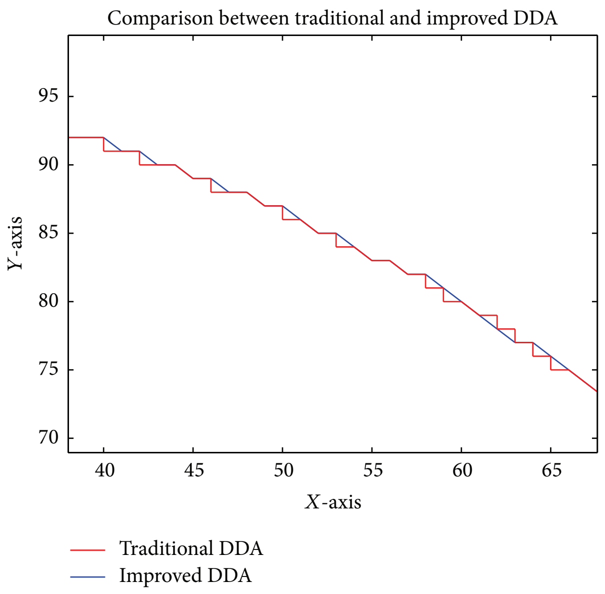

FIGURE 11: Detailed comparison between traditional and improved DDA.

of one axis has overflown, while the other is going to overflow in the next step.

Therefore, we can do the addition to the latter accumulator in advance in order to produce an advanced overflow. Thus, the recursion formulae can be rewritten as

$$\begin{aligned} x_{n+1} &= x_n + \mu_x IF\left(Ax_n + 2\lambda y_n > 2^n\right) \\ y_{n+1} &= y_n + \mu_y IF\left(Ay_n + 2\lambda x_n > 2^n\right) \\ Ax_{n+1} &= \left(Ax_n + y_n\right) \bmod 2^n \\ Ay_{n+1} &= \left(Ay_n + x_n\right) \bmod 2^n. \end{aligned} \tag{11}$$

As a result, the cursor will take one "combined" step instead of two separate steps, thus, eliminating the sawtooth in advance. With the same conditions as Figure 7, the simulation result of the improved DDA arc interpolation is shown in Figure 10. And the detailed comparison of the simulation results between the typical DDA algorithm and the improved one as shown in Figure 11, from which it is palpable that most of the sawteeth on the path are eliminated. We call this kind of algorithm improved DDA algorithm.

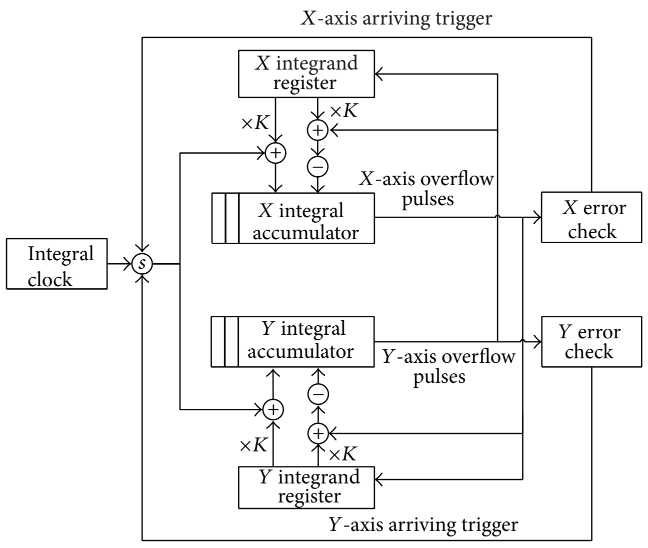

Figure 12: Logic structure of improved DDA arc interpolation.

In order to implement this improved DDA algorithm on FPGA device, the overflow conditions have to be changed, and the accumulator registers should add to 2 extra bits. One is to save the carry bit caused by overflow, while the other is sign bit, since the value of an accumulator can be negative. And the logic structure of the improved algorithm is shown in Figure 12.

In order to measure the performance of the two algorithms, we define path variance V_{arc} as

$$V_{\text{arc}} = \frac{\sum_{i=1}^{m} \left(\sqrt{x_i^2 + y_i^2} - R \right)^2}{m - 1}, \tag{12}$$

where R is the radius of the curve path and m stands for the number of the total interpolating steps from the start point to the end point.

For the typical DDA algorithm, with conditions in Figure 6, we have

$$V_{\text{arc}} = 15.6304. \tag{13}$$

While for the weighed DDA algorithm as shown in Figure 7, with the same conditions, we have

$$V_{\text{arc}} = 0.4223. \tag{14}$$

And for the sawteeth-eliminating DDA as shown in Figure 10, with the same conditions, we have

$$V_{\text{arc}} = 0.4051. \tag{15}$$

From the data, we can see that the improved DDA algorithm has the smallest path variance. Thus, it can be concluded that the performance of the sawteeth-eliminating DDA algorithm improved by us is better than the traditional one.

5. Electromagnetic Compatibility and Signal Integrity

5.1. Power Design

5.1.1. Power System. The maximum power allowed for a 32-bit PCI device is 25 W. And the power system of PCI bus is more complex than other kinds of buses, which has 6 different power connectors (+3.3 V, +5 V, +$V_{I/O}$, +12 V, −12 V, and +3.3 V_{aux}). Among them, +5 V, +$V_{I/O}$, +12 V, and −12 V are provided by system motherboard, while +3.3 V and +3.3 V_{aux} should be supplied from the device board, which means that a 5V-to-3.3 V power converting block is needed on board.

5.1.2. Power Layer. The device PCB board has four layers: top-layer, bottom-layer, power plane, and ground plane. Power plane, especially, should be divided into several power districts. If possible, high-speed signal wires will not go across two different power districts. Otherwise, adjust the direction of the slit to minimize the impact.

5.1.3. Power Decoupling. Every Vcc pin of every digital chip is assigned a decoupling capacitor connected to the ground. And every power pin is allocated a 0.047 μF electrolytic capacitor and a 0.01 μF nonpolar capacitor. What's more, pads and vias of the decoupling capacitors will not be 0.25 inches farther from corresponding Vcc pins or "golden-fingers," and routing wide shall be larger than 0.02 inches.

5.2. Signal Wire Routing

5.2.1. Clock Wire Routing. The clock signal of PCI device is based on reflected wave effect rather than incident wave effect. Therefore, the trace length of the PCI clock signal is 2.5 inches and ±0.1 inches for 32-bit PCI devices.

5.2.2. Nonclock Wire Routing. The maximum length of signal routing of 32-bit PCI slave device will not exceed 1.5 inches.

5.2.3. Pull-Up Resistors. Every control signal pin should be assigned with a pull-up resistor in case that these pins will not float when not driven. A PCI-slave-device developer need not care about this since it has been done on the motherboard of system.

5.3. Signal Integrity Test on PCI Pins. A simple series of tests on signal integrity of waveforms on PCI pins (the golden fingers) has been conducted on the PCI device board [22, 23]. The first test is to export a 20 MHz (20 Mhz is the maximum frequency that can be generated by the board, which is still close to 33 MHz, PCI clock signal) square wave to one golden finger from the board device and test it using an oscilloscope. And the test result is shown in Figure 13, from which it can be seen that the waveform is rather integrated, with steep rising edge and proper overshoot.

The second test is to export a 20 MHz square wave to one finger while testing that on an adjacent finger. This test is to judge how much interference one high frequency signal

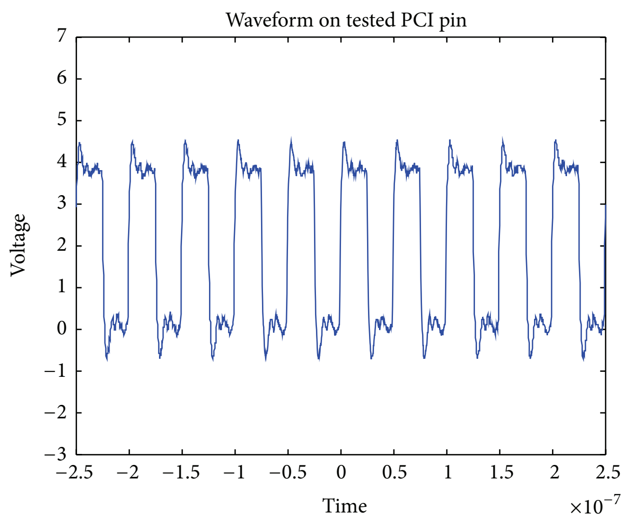

FIGURE 13: No. 1 SI test on PCI pins.

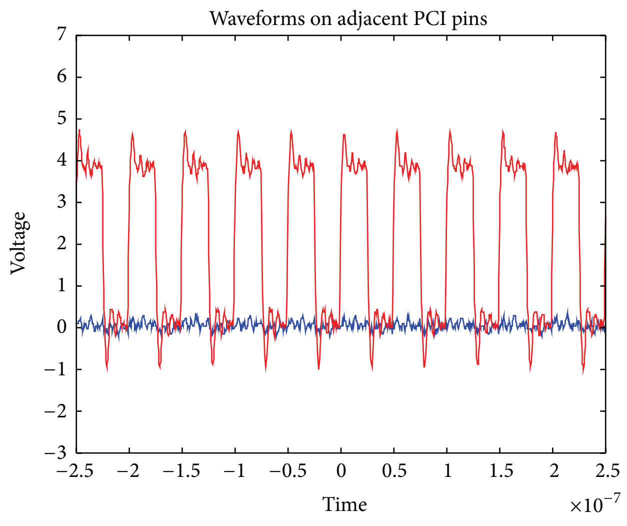

FIGURE 14: No. 2 SI test on PCI pins.

on one pin can cause on other PCI pins, especially on the adjacent ones. And the result is shown in Figure 14. It is clearly revealed that the waveform on the tested finger (the lower one) is much like that on the exported finger (the upper one), but the amplitude is far smaller, not enough to reach the threshold level. Therefore, it can be concluded that even though electromagnetic interference shows up on adjacent pins of the output, it does no harm to the functions of the system.

The third one is to export 2 waveforms to 2 adjacent fingers and observe both of them on signal integrity in order to measure the coupling interference. And the result is shown in Figure 15. It is obvious that both waveforms are highly integrated and without big distortion and interference. After the three tests, we can make a conclusion that the property of signal integrity of the board is rather remarkable. Thus, the harmful impact of electromagnetic interference is negligible.

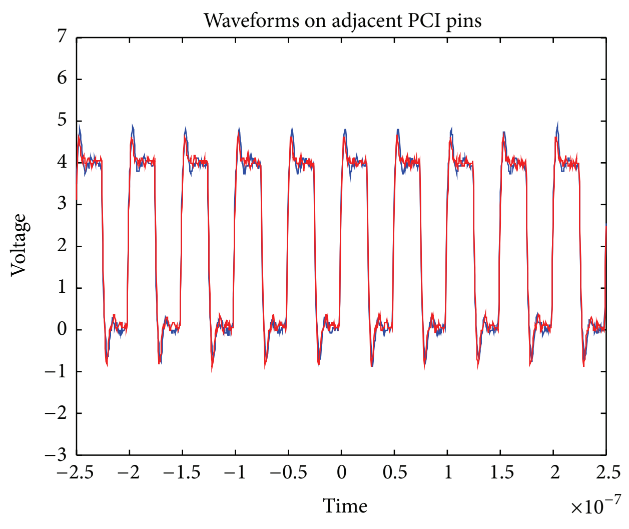

FIGURE 15: No. 3 SI test on PCI pins.

6. Conclusion

In this paper, a kind of motion control board is discussed. And the design scheme of the control board is reasonable and able to satisfy the requirements of biaxial motion systems. The PCI protocol decoding block is self-designed and functions well. And more importantly, we improved the typical DDA arc interpolation algorithm, broadened its application, and reduced its negative effect: the sawteeth. Currently, it has been put in use by our lab, and the result is rather remarkable.

Conflict of Interests

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

References

[1] W. Chen, Z. Wen, Z. Xu, and J. Wang, "Implementation of 2-axis circular interpolation for a FPGA-based 4-axis motion controller," in *Proceedings of the IEEE International Conference on Control and Automation (ICCA '07)*, pp. 600–605, Guangzhou, China, May-June 2007.

[2] F. Song, C. Wang, and S. Liu, "Designing of 8051 SC computer-based X-Y plotter control system," *Microcomputer Information*, vol. 23, no. 17, pp. 108–109, 2007.

[3] S. Zhang, Y. Zhao, Y. Fang, and X. Chen, "Design of the motion control system in chip mounter," *Manufacturing Informatization*, pp. 112–115, 2013.

[4] J.-F. Yan and N. Wu, "High speed DMA data transfer system based on PCI bus," *Journal of the University of Electronic Science and Technology of China*, vol. 36, no. 5, pp. 858–861, 2007.

[5] F. Zeng and S. Yao, "Design and application of open motion control system based on PCI bus," *Microcomputer Application*, vol. 32, no. 8, pp. 62–66, 2011.

[6] S. Yin, S. Ding, X. Xie, and H. Luo, "A review on basic data-driven approaches for industrial process monitoring," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 11, pp. 6418–6428, 2014.

[7] Y. Hua, Z. Jian-Min, and W. Mu-Lan, “Realization of DDA interpolation algorithm with FPGA device,” *Modern Manufacturing Engineering*, vol. 9, article 016, 2007.

[8] B.-T. Zhou and B.-J. Wang, “A DDA arc interpolator for digital differential analyzer based on FPGA,” *Electric Drive Automation*, vol. 5, article 004, 2005.

[9] S. Yin and G. Wang, “Robust PLS approach for KPI related prediction and diagnosis against outliers and missing data,” *International Journal of Systems Science*, vol. 45, no. 7, pp. 1375–1382, 2014.

[10] A. H. Khalil, M. A. Ashour, A. E. Salama, and H. I. Saleh, “FPGA implemented fast two’s complement serial-parallel multiplier with PCI interface,” in *Proceedings of the 10th International Conference on Microelectronics (ICM ’98)*, pp. 21–24, 1998.

[11] E. Finkelstein and S. Weiss, “PCI interface implementation using CPLD and FPGA devices,” in *Proceedings of the 9th Mediterranean Electrotechnical Conference (MELECON ’98)*, vol. 2, pp. 1284–1288, Tel-Aviv, Israel, May 1998.

[12] C. Jihai, Z. Ning, S. Lin, and Z. Xinchao, “Development of PCI high-speed data transmission board based on FPGA-rocket I/O,” in *Proceedings of the IEEE Conference on Electron Devices and Solid-State Circuits (EDSSC ’07)*, pp. 345–348, December 2007.

[13] G. Knittel, “A PCI-compatible FPGA-coprocessor for 2D/3D image processing,” in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 136–145, April 1996.

[14] Y. Mingji, W. Lei, and S. Haokun, “Design of PCI-104 bus interface of lowpower embedded CPU based on FPGA,” in *Proceedings of the International Conference on Measurement, Information and Control (ICMIC ’13)*, pp. 275–279, Harbin, China, August 2013.

[15] C. Zang and C. Shen, “Non-transparent PCI-to-PCI bridge based on verilog and FPGA,” in *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS ’06)*, pp. 2282–2285, Guilin, China, June 2006.

[16] S. Yin, X. Li, H. Gao, and O. Kaynak, “Data-based techniques focused on modern industry: an overview,” *IEEE Transactions on Industrial Electronics*, 2014.

[17] M.-J. Jeng, P.-Y. Kang, and Z.-H. Gao, “Implementation of multi-axis motion controller by FPGA-based hardware design,” *Advanced Science Letters*, vol. 19, no. 2, pp. 374–378, 2013.

[18] H. Qiu, C. Kai, and L. Yan, “Optimal circular arc interpolation for NC tool path generation in curve contour manufacturing,” *Computer Aided Design*, vol. 29, no. 11, pp. 751–760, 1997.

[19] S. Yin, X. Zhu, and H. Karimi, “Quality evaluation based on multivariate statistical methods,” *Mathematical Problems in Engineering*, vol. 2013, Article ID 639652, 10 pages, 2013.

[20] L. C. Wei, N. B. Z. Ali, and R. S. Nair, “Design of low cost FPGA based PCI bus Sniffer,” in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT ’03)*, pp. 420–423, Tokyo, Japan, December 2003.

[21] G. W. Vickers and C. Bradley, “Curved surface machining through circular arc interpolation,” *Computers in Industry*, vol. 19, no. 3, pp. 329–337, 1992.

[22] S. Yin and X. Gao, “Study on support vector machine based fault detection in Tennessee Eastman process,” *Abstract and Applied Analysis*, vol. 2014, Article ID 836895, 8 pages, 2014.

[23] Q. Bai, P. Jiang, and Y. Zuo, “VC implement of the DDA circular interpolation,” *Machine Tool and Hydraulics*, no. 7, pp. 219–220, 2006.