Development of a General Purpose PID Motion Controller Using a Field Programmable Gate Array

Sung-Su Kim* and Seul Jung**

Intelligent Systems and Emotional Engineering Lab.
Department of Mechatronics Engineering
Chungnam National University, Daejeon, Korea
(+82-42-821-6876, *a74110328@hanmail.net, **jungs@cnu.ac.kr)

Abstract: : In this paper, we have developed a general purpose motion controller using an FPGA(Field Programmable Gate Array). The multi-PID controllers on a single chip are implemented as a system-on-chip for multi-axis motion control. We also develop a PC GUI for an efficient interface control. Comparing with the commercial motion controller LM 629 it has multi-independent PID controllers so that it has several advantages such as space effectiveness, low cost and lower power consumption. In order to test the performance of the proposed controller, robot finger is controlled. The robot finger has three fingers with 2 joints each. Finger movements show that position tracking was very effective. Another experiment of balancing an inverted pendulum on a cart has been conducted to show the generality of the proposed FPGA PID controller. The controller has well maintained the balance of the pendulum.

Keywords: Motion controller, FPGA, VHDL, robot hand, inverted pendulum

1. INTRODUCTION

Since most of the actuators used in industries are dc/ac motors, the motion control technique is very important and become more integrated. In general, specialized commercial controller chips such as LM 629 that has a PID controller in it are used to drive motors. These chips may be used with/without a microprocessor. Programmable microprocessors are used most for controlling drivers of motors. However, most of microprocessors with controller chips have lack of fan-outs so that there is a limitation to be used in multi-channel systems due to slow calculation time of CPU or hardware limitation such as I/O pin number. These limitations lead to the use of a number of duplicate microprocessors or chips for controlling complicated system such as multi-joint robotic systems.

In our previous researches, we have used three 80196KC micro controllers and a serial communication box to control the robot hand. One microprocessor controls each finger with two actuators. In order to perform multi-serial communication with a PC, we needed a serial communication box. For encoder measurements, 6 counter chips such as LM 629 are used. These make the whole system bulky [1].

Here, we try to replace those massive duplicate hardware with an FPGA chip to make the whole control structure simple. The FPGA is a modifiable, flexible, and programmable device that can be implemented as many applied tool such as controllers, filters, computers, and etc [2-9].

We develop a general purpose motion controller using a field programmable gate array (FPGA). FPGA is known as a programmable hardware device that a user can design his/her algorithms and download it to a single chip. By using the high capacity of an FPGA, the additional hardware such as an encoder counter and a PWM generator, can be implemented in a single FPGA device. As a result, the noise and power dissipation problems can be minimized and it shows the cost effectiveness.

In this paper, we design PID control algorithm on a FPGA. The PID controller is a well known and easily implemented device for motion control applications. It works very well for linear systems with optimal gain tunings.

We also develop a PC GUI for an efficient interface control. The GUI has several functions such as setting of control parameters, downloading of trajectory, displaying the control states, and so on. A user can specify PID controller gains through GUI program in PC. Those gains are transmitted though a serial communication to a microprocessor. And a microprocessor sends them to a FPGA. The purpose of a microprocessor here is to make communication between a FPGA and a PC possible.

The total controller size and power dissipation are effectively reduced by developing a one-chip motion controller using FPGA. We test the FPGA based PID controller by controlling a robot finger and an inverted pendulum.

The proposed controller is required to control the balance of the pendulum and the position tracking of the cart simultaneously. Performances of position tracking of the cart while balancing the pendulum were successfully achieved.

2. OVERALL SYSTEM STRUCTURE

The overall system block diagram of the robot hand control is shown in Figure 1. It consists of robot fingers, an FPGA controller, a motor driver, an 80196 microprocessor, and a PC.

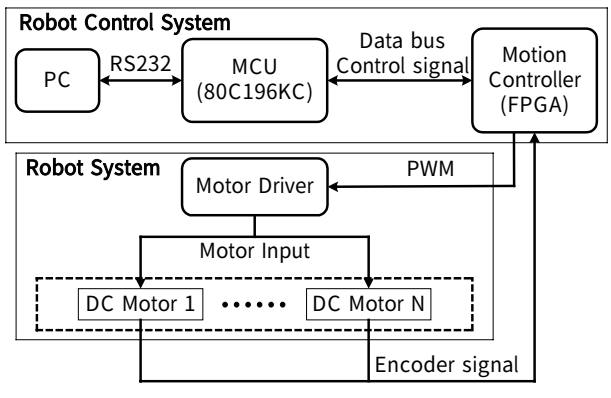

Fig. 1 Overall structure of robot hand control

Different from the previous model [1], an FPGA replaces a communication box as well as several MCUs.

The PC communicates with an 80196KC microprocessor through a serial communication for user's convenience.
The 80196 microprocessor communicates with an FPGA chip to transfer desired trajectories and controller gains at each sampling time. The FPGA chip programmed for PID controller generates PWM signals to the motor drivers. Then each motor is actuated and each motor's encoder measurement is transferred to the FPGA chip for making positional errors.

3. GUI

The interfacing GUI program helps users to manipulate controllers' gains easily and to observe the response of each motor. Figure 2 shows the GUI window that has many functions.

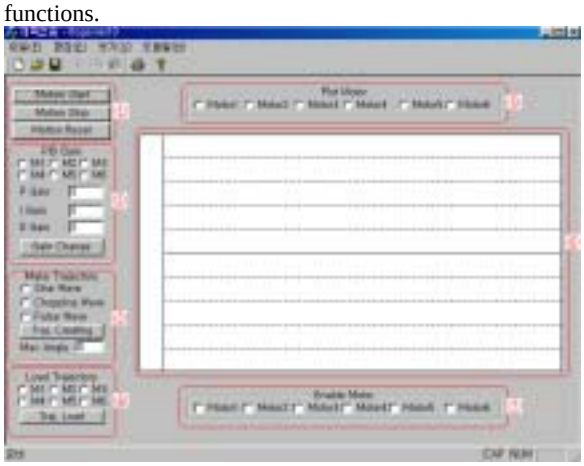

Fig.2 GUI

Each part has the following function:

① Motion control part(start, stop)
② PID gain selection part
③ Trajectory planner
④ Download trajectories
⑤ Selection of motors to plot
⑥ Display panel
⑦ Motor selection

4. PID CONTROLLER ON FPGA

4.1 Overall structure

The PID controllers are implemented in an FPGA chip having 300,000gates made from Altera Company.

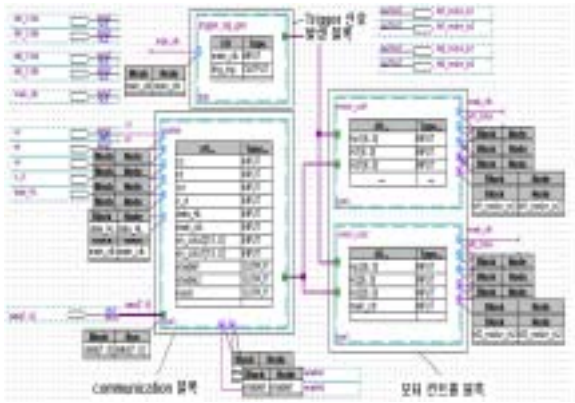

Figure 3. The design of motion controller on Quartus II 2.0

The design of PID motion controller is programmed by Quartus II 2.0. Figure 3 shows the schematic design of the controller. The FPGA based PID controller consists of several blocks such as a communication block, an encoder counter block, a PID calculation block, and a PWM generation block.

Table 1 lists functions of each pin shown in figure 3.

Table 1. The pin name and its function

	Pin name	functions
Input	M1_CHA	Motor 1 encoder phase A
	M1_CHB	Motor 1 encoder phase B
	M2_CHA	Motor 2 encoder phase A
	M2_CHB	Motor 2 encoder phase B
	main_clk	FPGA system clock
	Cs	Chip select
	Rd	Read
	Wr	Write
	c_d	Command, data
	data_HL	High, low byte
Output	M1_motor_in1	Motor 1 output 1
	M1_motor_in2	Motor 1 output 2
	M2_motor_in1	Motor 2 output 1
	M2_motor_in2	Motor 2 output 2
	enable1	Motor 1 enable
	enable2	Motor 2 enable
Bidirection	data[7..0]	8bits data bus

The detailed function of internal blocks of the FPGA is shown next.

4.2 The communication block

It receives PID gains and desired trajectory, and transfers encoder data from the controller to an MCU whenever they are needed. Data write from an MCU to the motion controller includes PID gains, enable/disable, the motion controller reset. Table 2 shows commands and their data.
Data read from the motion controller to an MCU is to read encoder data. Since the size of encoder data is 16 bit, it should be read as high and low bytes separately.

Table 2. The pin name and its function

Command	**Data**	**Command**	**Data**
Motor 1 Kp	0x01	Motor 2 Kp	0x11
Motor 1 Ki	0x02	Motor 2 Ki	0x12
Motor 1 Kd	0x03	Motor 2 Kd	0x13
Motor 1 Yd	0x04	Motor 2 Yd	0x14
Motor 1 enable	0x05	Motor 2 enable	0x15
Motor 1 disable	0x06	Motor 2 disable	0x16
Read encoder Motor 1	0x07	Read encoder Motor 2	0x17
FPGA reset	0x08		

4.3 Motor control block

Figure 4 shows the motor control block. It consists of encoder counter, PID controller, and PWM generator. The

clock synchronizes the process. It generates the trigger signal at each 1kHz sampling time and sends it to encoder counter block and PID controller block to synchronize the process of encoder counter and PID controller.

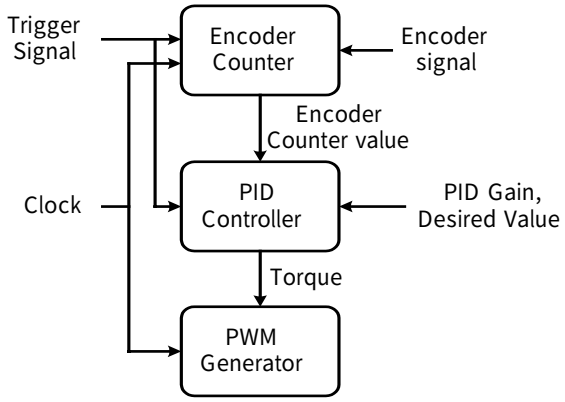

Figure 4. The structure of motor control block

Figure 5 shows the designed motor control symbol. It takes PID gains, a clock, a trigger signal, encoder signals, a reset signal as input and pwm signal and encoder counter values as output

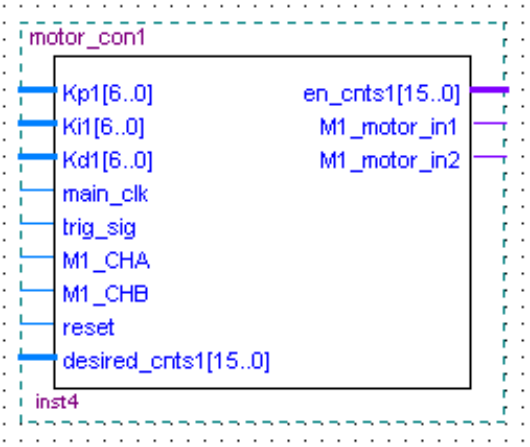

Figure 5. The designed symbol of motor control block

4.3.1. Encoder Counter Block

It counts and determines the direction of motor rotation from encoder signals. The block diagram is shown in Figure 6. Difference in phase A and B determines the direction of rotation. Noise from a mechanical system can be filtered out by a digital filter as shown in figure 7. Every trigger signal enables to generate counter values. 16 bit of counter limits the range of the movements.

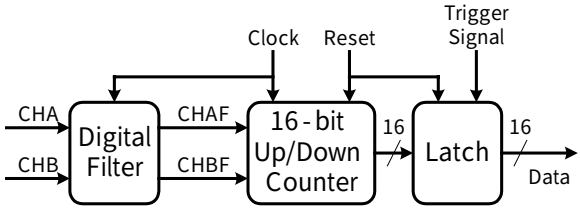

Figure 6. The structure of encoder counter

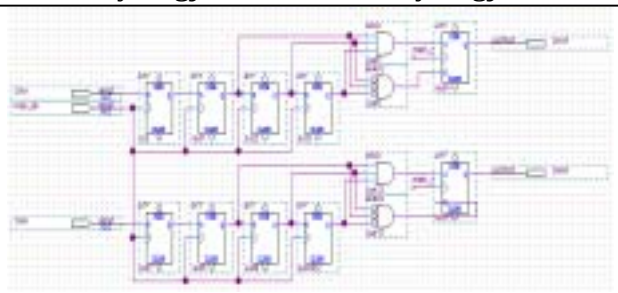

Figure 7. The digital filter of encoder counter block

4.3.2. PID Controller block

It receives encoder data from the encoder counter block and compares them with desired values to generate positional errors and then generates PID control torques. PID control equations are

$$\tau(n) = K_p e(n) + K_i s(n) + K_d \{e(n) - e(n-1)\} \quad (1)$$

$$s(n) = \begin{cases} s_i & \sum e(n) > s_i \\ \sum e(n) & -s_i \le \sum e(n) \le s_i \\ -s_i & \sum e(n) < -s_i \end{cases} \quad (2)$$

$\tau(n)$ is control input, $e(n)$ is error, K_p, K_i, K_d are PID gains, s_i is a threshold value.

.

4.3.3. PWM Generator block

The below is PWM generation program written in VHDL. It's frequency is 12 ㎑.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY pwm_generator IS
    PORT
    (   main_clk : IN STD_LOGIC;
        pwm_mag : IN integer range 0 to 255;
        pwm_signal : OUT STD_LOGIC  );
END pwm_generator;
ARCHITECTURE pwm_generator_architecture OF pwm_generator IS
    signal      pwm_count   :   integer range 0 to 256*8-1;
    signal      pwm_buff    :   std_logic;
BEGIN
    process(main_clk)
        variable    mag :   integer range 0 to 256*8-1;
    begin
        if main_clk = '1' and main_clk'event then
            mag := pwm_mag * 8;
            if pwm_count >= mag then
                pwm_buff <= '0';
            else
                pwm_buff <= '1';
            end if;
            pwm_count <= pwm_count + 1;
            pwm_signal <= pwm_buff;
        end if;
    end process;
END pwm_generator_architecture;
```

Figure 8. VHDL code of PWM generator block

Motion controller is designed by Quartus II 2.0, and implemented on APEX II Series EP20K300EQC240 by Altera company.

Processing status	Fitting Successful
Chip name	motion_controller
Device name	EP20K300EQC240-2
Total logic elements	2236 / 11520 (19 %)
Total pins	24 / 157 (15 %)
Total ESB bits	0 / 147456 (0 %)

Figure 9. Whole resources amount of used FPGA

6. EXPERIMENTS

6.1. Robot hand

Figure 11 shows the real experiment setups. The robot hand has three fingers and each finger has three joints, but only two joints are actuated. The FPGA controller controls the movements of the robot hand. Movements are displayed on the screen.

Figure 11. Finger robot system

Experiments of grasping an object are done. Each finger is commanded to move certain degrees to make grasping. Figures below show encoder values of the movements of robot finger. PID controller gains for upper joints of each finger are 1.875, 0.04, 1.875, respectively. For lower joints, PID gains are 0.6, 0.007, 0.6.

Figure 12 show the position tracking of upper joint of the first joint.

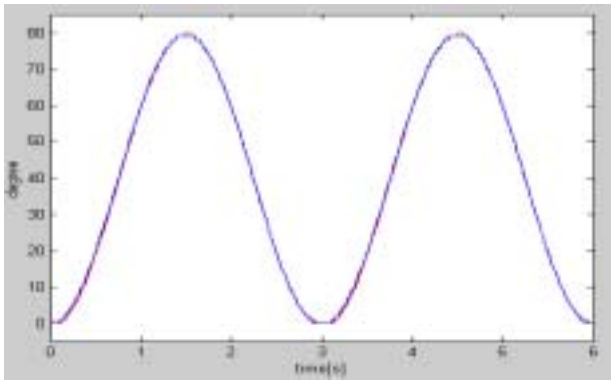

Figure 12. The trajectory of upper joint of first finger

Figures 13-17 show the position tracking of each joint of the robot. We see that tracking is well done.

Figure 13. The trajectory of upper joint of second finger

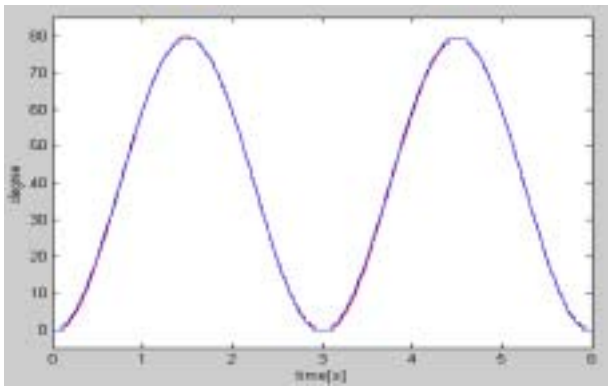

Figure 14. The trajectory of upper joint of third finger

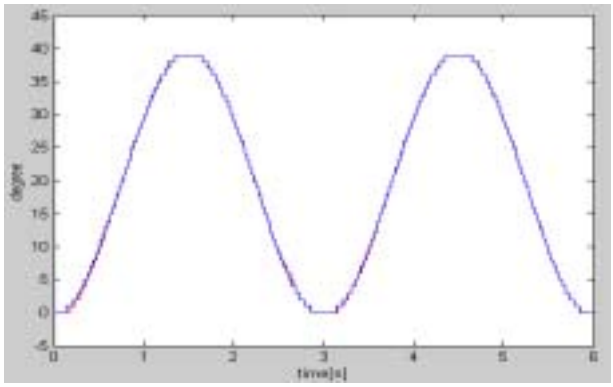

Figure 15. The trajectory of lower joint of first finger

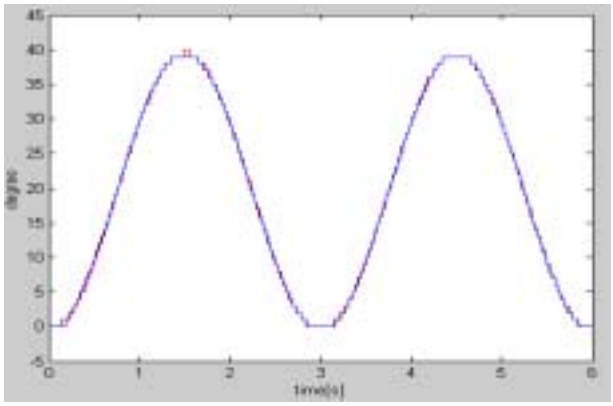

Figure 16. The trajectory of lower joint of second finger

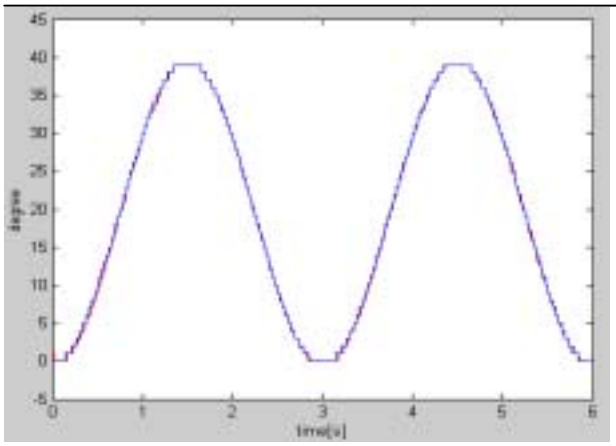

Figure 17. The trajectory of lower joint of third finger

In order to see the robot hand movement to grasp an object, combine all the plots together. Combining all encoder values of the robot hand movements yields the plot shown in figure 18. The robot hand moves from initial position to grasp the object.

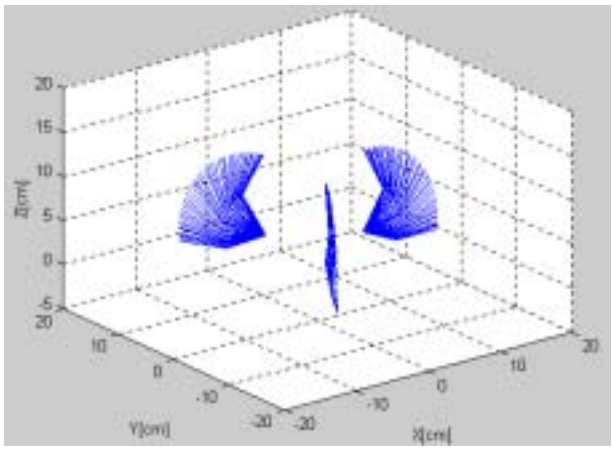

Figure 18. The movement of finger robot in 3-dim space

Another experiment of tracking sinusoidal function trajectory with different frequency of 1 ㎐ is done.

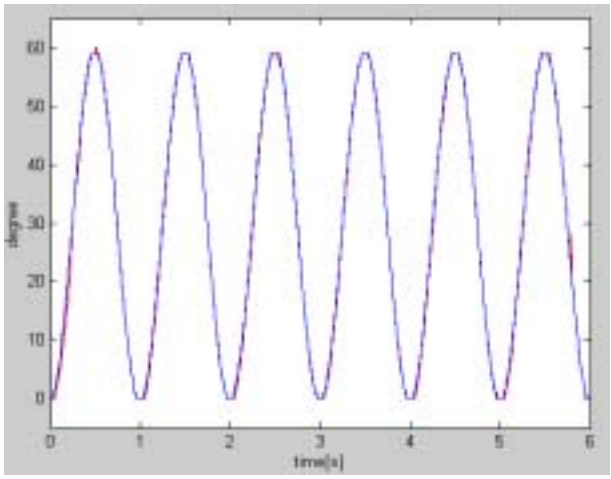

Figure 19. The trajectory of upper joint, 1.000 ㎐

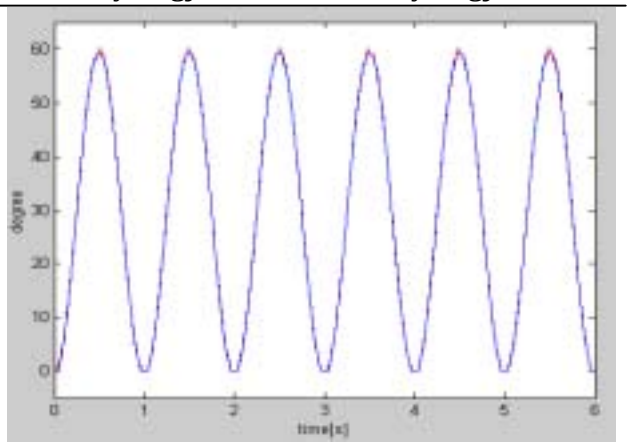

Figure 20. The trajectory lower joint, 1.000 ㎐

2 Inverted pendulum

Next experiment is to control an inverted pendulum by the same controller. Figure 21 shows the setups of the experiment.

Figure 21. Inverted pendulum system

The angle of the pendulum is shown in figure 22. The pendulum is well maintained balancing. Oscillation range is about 0.4 degrees since the cart is moving

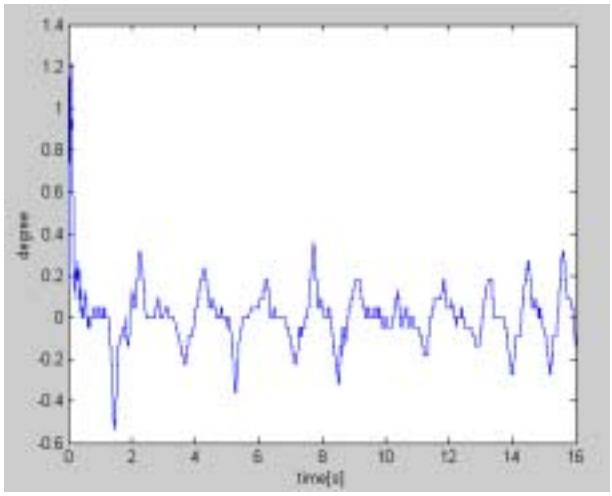

Figure 22. The angle of inverted pendulum

The corresponding cart movement is plotted in figure 23.

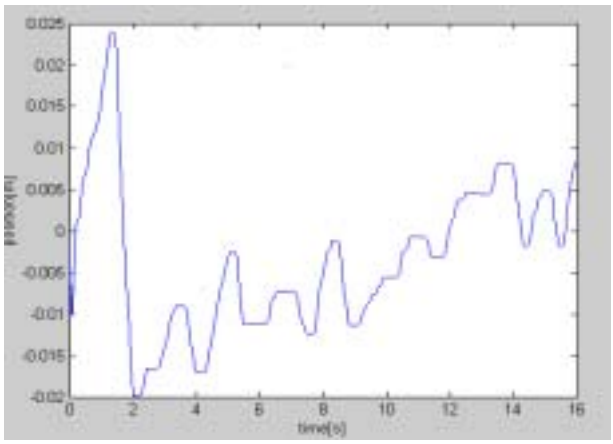

Figure 23. The position of a cart

The pendulum has successfully maintained balancing within $\pm$0.4 degree. However, the cart position oscillates within $\pm$ 0.02m, and keeps moving toward one direction This is due to nonlinear behavior of the system. This problem can be solved by using an intelligent controller [10].

6. CONCLUSIONS

This paper addresses the hardware implementation of an FPGA based PID controller. The proposed controller was tested by performing control of a robot hand and an inverted pendulum. The PID controller for each finger joint is implemented by an FPGA. Control using an FPGA was successful and joint movements are displayed on GUI.

Implementing PID controllers with FPGA turned out to be very effective since the size of the whole system was reduced tremendously and it was cost effective as well. Since the finger is actuated at the joint directly, smaller design of linkages and connectors were very difficult to meet very accurate specification. The implemented controller was successfully tested for controlling an inverted pendulum.

REFERENCES

[1] Jeonggu Kim, Sung Su Kim & Seul Jung, “Implementation and Motion Control of Linkage Bar Fingers”, pp. 2091-2094, ICCAS 2002

[2] T. Aoyama, Qianyi Wang, R. Suematsu, R. Shimizu, U. Nagashima, “Learning algorithms for a neural network in FPGA”, *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 1, pp. 1007 –1012, 2002.

[3] M. Krips, T. Lammert, A. Kummert, ”FPGA implementation of a neural network for a real-time hand tracking system”, *Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications*, pp. 313 –317, 2002.

[4] M. Cristea, J. Khor, M. McCormick; “FPGA fuzzy logic controller for variable speed generators”**,** *Proceedings of the 2001 IEEE International Conference on Control Applications*, pp. 301 –304, 2001.

[5] Abdelkrim K. Oudjida et al, “A reconfigurable counter controller for digital motion control application”, *Microelectronics Journal*, vol. 28, no. 6-7, 1997.

[6] F. Thomas et al, “Design and implementation of a wheel speed measurement circuit using field programmable gate arrays in a spacecraft”, *Microprocessors and Microsystems*, pp. 553-560, 1999.

[7] S. N. Oh et al, "Design of a biped robot using DSP and FPGA", *Proceeding 2002 FIRA Robot World Congress*. 698-701.

[8] A. Kongmunvattana, P. Chongstivatana, “A FPGA-based Behavioral Control System for a Mobile Robot”, *The 1998 IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 759-762, 24-27, Nov, 1998,

[9] "LM628/LM629 Precision Motion Controller", *National Semiconductor Corporation*, Nov, 1999.

[10] Sung-su Kim and Seul Jung, “Implementation of an intelligent controller with a DSP and an FPGA for Nonlinear systems” to appear in ICCAS 2003.