The Modelling, Simulation and Hardware Implementation for FPGA-based Stepping Motor Motion Drives

Chiu-Keng Lai, Wei-Nan Chien, Shou-Liang Tsai and Yaw-Ting Tsao

Abstract

In this study, we establish the model and implement the hardware controller for the stepping motor drive system using FPGA. Through the full bridge power circuits for a hybrid stepping motor, we create two Verilog HDL-based systems by peak current and average current methods to simulate and realize hardware the current regulators. The hardware systems are first simulated with ModelSim and Quartus II, and the resulting Verilog HDL codes are implemented on Altera FPGA as current regulators. The hardware systems are first simulated with ModelSim and Quartus II, and the resulting Verilog HDL codes are implemented on Altera FPGA as current regulators to adjust the duty cycle to regulate the stepping motor current. Human machine interface and motion control are considered to the experiment simultaneously. The performances for the two proposed hardware controllers are compared.

Keywords: FPGA, Stepping Motor, Peak Current Regulation, Average Current Regulation, Matlab/Simulink, Modelsim.

1. Introduction

Field Programmable Gate Arrays (FPGAs) allow designers to integrate their hardware functions in a chip. The researches of FPGA applied to the controller design have been reported, and such papers could be found in [1-4]. Chan, Mouleem and Wang designed their PID controller by FPGA, and applied it to realize a temperature controller [1]. Tezuka, Ichikawa and Noda designed and implemented their motor tracking controller on FPGA using two-degree-of-freedom control strategy [2]. Siwakoti and Town used Matlab/Simulink to simulate and develop their motor drive and PWM control module [3]. Finally, Gao and Kennel implemented a hybrid sensorless control for SMPMSM using FPGA [4].

Regarding the stepping motor drive design realized by FPGA, several reports are seen in [5-9]. In [5], the authors used Cyclone II FPGA and two digital-to-analog converters (DACs) to implement the micro-stepping control, where velocity and position profiles are generated by FPGA. In [6], the authors adopted analog to digital converter (ADC) to obtain

In this paper, we focus the development of a hybrid stepping motor drive system on current regulator design, drive system modelling, performance simulation, functional analysis and hardware implementation. We first build a stepping motor current regulation model for simulation by Matlab/Simulink and ModelSim system. The developed system is subjected to the following limitations: (1) The trapezoidal velocity profile is for motion control. (2) The maximum velocity is limited by the desired position displacement. (3) The current regulating strategies are, respectively, the peak current method and the average current one. Regarding the hardware implementation, in order to leave out the usage of hall current sensors and ADC converters, a one-bit comparator is used to detect the winding current, and its output is fed back to the controller. Further, to get the average current without the ADC, we develop a hardware system by FPGA to obtain the average current. Combining several blocks, the two proposed FPGA-based digital controllers are then synthesized to simulate the current control on Quartus II and ModelSim. Both the regulation methods are simulated by different speeds and their results are compared.

Based on the created hardware systems by Matlab/Simulink, and through some modifications to fit the Altera Cyclone III C10 FPGA, the hardware regulators are finally realized on Altera Cyclone III FPGA, and are applied to hybrid stepping motor drives to practically realize the current regulation according to the desired velocity profile. At last, the simulations and experimental results are verified and compared to each other.

The paper is organized in the following manner. The hardware drive systems for a hybrid stepping motor are first shown in Section 2. Next, Section 3 describes the algorithms of current regulation about the methods of peak current and average current by a signal flow chart. And Section 4 shows the simulated results for both the proposed structures with Quartus II and ModelSim. Then the experimental setup and results of the two proposed methods are illustrated and compared in Section 5. And finally, the conclusions are given in Section 6.
2. The Hardware Circuits of a Stepping Motor Drive System

The circuits used to the modelling and hardware implementation for stepping motors include FPGA, gate driver, power converter and current sensor. The FPGA-based control board designed for the hardware controller is shown in Fig. 1, which is mainly realized on Altera Cyclone III C10 FPGA. Besides, an 8051 compatible micro-controller IP is embedded in the FPGA to do the following things: (1) to help the realization of pulse width modulation (PWM); (2) to realize the digital differential analyzer (DDA) and acceleration and deceleration control under a trapezoidal velocity profile, (3) and to be as the interface for the communication between current regulator of the study and human machine interface (HMI) on PC. Besides, due to the purposes of hardware modelling, hardware and software co-simulation and practical hardware implementation, the blocks of PWM and current control are further considered and discussed in the next section.

The circuit of power converter for a hybrid stepping motor drive system is shown in Fig. 2. The DC bus voltage is 48 V, and current command is set as 1 A. The stepping motors are operated in a full-step mode, and winding current detection is executed by resistors $R_A$ and $R_B$. The voltages on resistors are proportional to the winding currents, and are compared with the desired current level by two analog comparators. The output signals of comparators, $V_A$ and $V_B$, are then fed back to the current regulators established by FPGA to make the current reach the desired level.

3. The Algorithms of Current Regulation

The two current regulating algorithms of peak current and average current are applied to the current control of stepping motor drive design. The current regulating sequence is done by three modules: dead-time module, delay-time module and PWM module. Dead-time module is used to set the switching dead time to be 1 $\mu$s, while delay-time and PWM modules are designed to make the winding current reach to the desired level rapidly, and keep on it consistently until the next excitation phase is alerted. The current control is implemented on 20 kHz sampling frequency. The inputs of current comparator are, respectively, from the command of winding current and the detected voltage on $R_A$ and $R_B$. The two proposed hardware control algorithms are implemented on FPGA and leave out the necessary ADC converter. The procedures are shown in the following.

3.1 The Peak Current Regulation

The peak current regulating procedure runs with a time period of 50 $\mu$s, or 20 kHz sampling frequency. For a new stepping command arrival, the MOSFET switch is first turned on, and the corresponding phase current increases. In the moment, a 4 MHz sampling clock monitors the output of the current comparator to decide the conditions of switches. The strategies are: if the level of winding current ($I_{fb}$) is lower than the desired current command, $I_{cmd}$, i.e., the comparator outputs $V_A = 0$. The MOSFET switch keeps turning on. Once the level of winding current is greater than the command, i.e. $I_{fb} > I_{cmd}$ or $V_A = 1$, the MOSFET switch is then turned off, and the motor current decreases. The switches keep off until the next PWM cycle alerts. The flow chart of peak current regulation is shown in Fig. 3.
3.2 The Average Current Regulation

In this strategy, the delay-time and PWM modules are used, and the average current regulation procedure runs on a period of $50 \mu s$ as the same peak current strategy. The concept is based on the average of winding currents during the sampling period to get the average current without using ADC, so it is the most important design. The flow chart of average current regulation is shown in Fig. 4, and the details are as follows.
First, the current detector checks whether the current feedback signal \( I_{fb} \) exists or not. If no feedback signal is detected, the PWM module then sets the duty cycle to 0 %, i.e., turns off the PWM modules. Otherwise, if the current feedback signal exists, the PWM duty cycle is then set to be 100 %. In this during, it is called the delay time, and the winding current increases rapidly. After the comparator outputs \( V_A = 1 \), i.e., \( I_{fb} \geq I_{cmd} \), the PWM module then resets the PWM duty cycle to 50 %, and adjusts the duty cycle according to the average of winding current. In the developed system, the upper and lower bounds of PWM duty cycle are set to be 80 % and 20 %, respectively. The adjusting procedures of PWM duty cycle for average current regulation are: (1) If the winding current is greater than the current command \( I_{fb} > I_{cmd} \), the PWM module decreases the PWM duty cycle 1 % per sampling period (0.25 \( \mu s \)), and the minimum duty cycle is 20 %. (2) If the average current is smaller than the current command, then PWM module increases the PWM duty cycle 1 %, and the maximum duty cycle is 80 %.

In this strategy, the average of winding currents is calculated by a counter which accumulates the amount of clock pulses for the current level being higher or lower than the command, and the frequency of clock pulse is 4 MHz. When the output of comparator is \( V_A = 0 \), the counter increases, otherwise, the counter decreases. If the accumulated value is negative, the average of winding currents is smaller than the desired. Otherwise, the average current is greater than the desired. With this algorithm, the ADC for current detection is unnecessary.

4. The System Modelling and Simulation

The overall block diagram for simulations shown in Fig. 5 includes all the components corresponding to the real hardware system. In Fig. 5, the block (1) includes the generation of speed command and DDA algorithm, and block (2) is the current regulator; block (3) is the driver and motors, and block (4) is the current comparators. Motion control is dominated by a trapezoidal velocity profile as shown in Fig. 6. Two-axis synchronous motion control is realized by DDA and included in block (1) [10]. The maximum operation speed is set as 180 rpm which is according to the real condition for a commercial caving machine [10].

The stepping motor model is created by Sim Power System module in Matlab as shown in Fig. 7. Furthermore, the current control module is shown in Fig. 8, which is developed to generate the corresponding Verilog HDL file for realizing the actual hardware controller by FPGA. In this module, two 0.5 \( \Omega \) resistors are set to detect the current as in the real system.
To verify the accuracy of the created modelling system, simulations and practical experiments are done. The simulations are carried out based on the Matlab/Simulink and Modelsim system as shown in Fig. 5, and the stepping motor is set on the full-step mode. The speed commands and DDA system are operated under period of 6.5 ms, and maximum 4 steps per sampling time are generated as shown in Fig. 6; the corresponding speeds are approximated as 45 rpm, 90 rpm, 135 rpm and 180 rpm. The simulated results for the proposed two current regulation methods are demonstrated in the following.
First, to demonstrate the four speed commands shown in Fig. 6 in steady-state condition, Figs. 9 and 10 are the results to demonstrate their performances. In Fig. 9, it shows the four step speeds, 45 rpm, 90 rpm, 135 rpm and 180 rpm in peak current regulation, and the system which is modulated in 20 kHz has lower ripple current as compared with those results shown in Fig. 10 which is from the simulated results of the average current regulation. Since the adjusting rate for the average current method is $\pm 1\%$ per sampling time, the slow adjusting rate causes the system with a slow regulation rate in their current response as compared with the method of peak current regulation. Regarding the acceleration and
deceleration conditions for the two methods, Figs. 11 and 12 illustrate their results with the constraints as shown in Fig. 6. Fig. 11 gives the results for peak current regulation in acceleration and deceleration conditions, and Fig. 12 is the corresponding results of average current strategy.

Figure 9: The steady-state current responses for peak current control method for speed command: (a) 45 rpm, (b) 90 rpm, (c) 135 rpm and (d) 180 rpm.

Figure 10: The simulated steady-state currents respond to the average current control method for speed commands: (a) 45 rpm, (b) 90 rpm, (c) 135 rpm and (d) 180 rpm.
Comparing the four simulated results, owing to the slow change rate of duty cycle, the average current regulation has a bigger current ripple than the results from the method of peak current regulation. However, the average current method has a smaller noise component when compared with the peak current method. For average current regulation, to increase the adjusting rate, such as ±2%/s or even bigger, it may lead to the current response similar to a PI controller with a high proportional gain. If the adjusting rate is large enough, the responses controlled by the average current regulation will approach to the results by the peak current regulation method.
Figure 15: The one-axis trigger signals measured by oscilloscope.

Figure 16: The experimental results of the gate driver signals and current responses in acceleration and deceleration. (a) The peak current regulation algorithm; (b) the average current regulation algorithm.
5. The Experimental Setup and Results

To actually verify the validness of system modelling and hardware controller design, an XYZ-table which is driven by stepping motors is built as shown in Fig. 13. It includes the human machine interface (HMI), the power converter, the speed command generator, comparator and current regulator as the block diagram shown in Fig. 5. HMI programmed by C++ language can directly read the G codes and convert them into a series of stepping commands. The stepping commands on HMI are transferred to the stepping motor drive system through the UART interface. The stepping motor drives are controlled by FPGA-based system, and the hardware Verilog code is created from those modelling as stated in Section 4, and realized by Altera Cyclone III C10 FPGA. The power converter includes the photocouples, gate drivers IR2101 and power MOSFET. The experimental current responses are captured by a current probe and digital oscilloscope to illustrate their performances.

The procedures of experiment are as follows:

1) The controller is designed to drive the XYZ-table for a commercial caving machine. The stepping commands are from the G-code on HMI, and they are transferred to the FPGA-based controller by UART interface with 19200 bps.

2) The acceleration and deceleration are according to the desired trapezoidal velocity profile as shown in Fig. 6. The stepping commands and the constraints of maximum speed are also decided by the HMI system.

3) An embedded micro controller 8051 in FPGA receives the serial commands from HMI, and converts them into the stepping commands for 3-axis stepping motor drives. According to the stepping commands, the designed three control modules: dead-time module, delay-time module and PWM module, are then sequentially executed to control the currents on the desired level. The experimental results about the hardware regulators are shown in the following.

To verify the validness and correctness, the conditions of hardware realizations are set the same as the software evaluation process as shown in Section 4. The PWM signals of power converter are captured by a logic analyzer and digital oscilloscope as shown in Figs. 14 and 15, respectively. It is obvious that the current control process includes the dead-time module, delay-time module and PWM module.

First, the effects of acceleration and deceleration for both the current control strategies are shown in Fig. 16, where there are the current responses from the hardware realized by a peak current regulator. The traces are the PWM signals of gate driver and the corresponding phase current, respectively. Besides, Fig. 16(b) shows the results of a current control by the average current regulator. The two figures are very similar to the results as the simulations by Matlab/Simulink and ModelSim. The peak current response is fast enough to regulate the stator current to a quite small ripple as compared with the average current algorithm. However, the peak current strategy has a bigger high frequency noise than average current one.

Afterward, the following demonstrations are the four steady-state current responses of 45 rpm, 90 rpm, 135 rpm and 180 rpm, respectively, to verify the performance in different operating speeds. Figs. 17(a)~(d) are the results for motors controlled in peak current regulation, and Fig. 18(a)~(d) are the corresponding results for average current regulation. These responses and trajectories deeply match to the co-simulation results as shown in Section 4. Through those simulation results and hardware realized system, the modelling about the stepping motor drive system incorported with an XYZ-table and the hardware controller created by Verilog HDL codes are successfully modelled and hardware realized.
Figure 17: The steady-state current responses for peak current control method on speed command: (a) 45 rpm, (b) 90 rpm, (c) 135 rpm and (d) 180 rpm.
Figure 18: The steady-state current responses for average current control method on speed command: (a) 45 rpm, (b) 90 rpm, (c) 135 rpm and (d) 180 rpm.
6. Conclusion

In the paper, we have proposed two hardware implemented current regulators by peak current and average current methods for stepping motor drive systems. The designed systems are first modeled by Matlab/Simulink and Modelsim. The proposed systems are first evaluated by simulations with the co-design including the Verilog HDL codes and Matlab/Simulink system. And the cores of hardware regulators are implemented on an Altera Cyclone III FPGA to verify the performances. Co-simulation and hardware realized controllers show that the proposed modelling systems and their hardware design methodologies can be easily used to help the user evaluate their developing system on the hardware functions and the system’s performances when FPGA are used to the digital controller design.

Acknowledgement

The authors would like to thank the financial support by the project no. NCUT201314 from the National Chin-Yi University of Technology.

References


Chiu-Keng Lai received the B.S. degree in electronic engineering from National Taiwan University of Science and Technology, Taipei, Taiwan in 1989, and the M.S. and Ph.D. degrees in electrical engineering from National Central University, Chung-Li, Taiwan, in 1991 and 2001, respectively. In 1992, he joined National Chin-Yi University of Technology, Taichung, Taiwan, where he is an Associate Professor in the Department of Electrical Engineering. His teaching and research interests include variable structure system designs and applications for motor drive systems, PC-based control system designs with LabVIEW and Matlab/Simulink, and digital circuit
designs and applications with FPGAs on motor drive systems and motion controls.

**Wei-Nan Chien** received the B.S. and M.S. degrees from National Chin-Yi University of Technology, Taichung, Taiwan in 2011 and 2014 respectively, all in electrical engineering.

**Shou-Liang Tsai** received the B.S. and M.S. degrees from National Chin-Yi University of Technology, Taichung, Taiwan in 2011 and 2013 respectively, all in electrical engineering. He is now with the Hiwin Mikrosystem Corp. in Taichung, Taiwan as an engineer.

**Yaw-Ting Taso** received the B.S. and M.S. degrees from National Chin-Yi University of Technology, Taichung, Taiwan in 2012 and 2015 respectively, all in electrical engineering.