TELKOMNIKA, Vol.11, No.1, January 2013, pp. 279~286
ISSN: 2302-4046

The Implementation of S-curve Acceleration and Deceleration Using FPGA

Guangyou Yang*[1,2], Zhijian Ye[1,2], Yurong Pan[1,2] and Zhiyan Ma[1,2]
[1]School of Mechanical Engineering, Hubei University of Technology, Wuhan, China
[2]State Key Lab. Of Digital Manufacturing Equipment & Technology (HUST), Wuhan, China
*corresponding author, e-mail: pekka@126.com

Abstract

This paper analyzes the basic principle of S-curve acceleration and deceleration, and presents an implementation method of S-curve control algorithm based on FPGA. The S-curve function module diagram and implementing method of PWM speed adjusting are described in FPGA. The steps of S-curve speed dissociation module and PWM variable frequency speed control are shown in this paper. High Performance and Capacity Mixed HDL Simulation software – ModelSim is used to verify the algorithm Verilog HDL code executed in FPGA. At last, a test experiment utilizing such algorithm mentioned above is carried out on one x-y working stand. There is a good agreement between simulation and experiment. The experimental results indicate that the algorithm is simple and reliable enough to meet different application requirements.

Keyword: *FPGA; S-curve; Acceleration/deceleration; PWM.*

1. Introduction

The speed control of motor acceleration and deceleration is a crucial factor when high speed and precision position of motor is needed [1], [2]. The common speed controlling curves are linear, index and S-curve. The flexibility of control system with linear speed control is poor. It triggers potential problems such as shock and acceleration/deceleration mutations when the motor start up and acceleration/deceleration phase is over. Therefore, this algorithm is not suited for a control system which requires high speed and precision. By contrast, the S-curve algorithm is usually used in many high grade multi-axis motion control systems. It reduces shock and makes full use of the motor performance by attenuation of the motor deceleration in start-up stage [3]. In order to improve the real-time performance of motion control system, the acceleration/deceleration speed control algorithm must be simple [4]. This paper analyzes S-curve speed control algorithm and realization in FPGA.

2. System structure of S-curve acceleration and deceleration based on FPGA

System structure of S-curve method implemented in FPGA is shown as Figure 1.

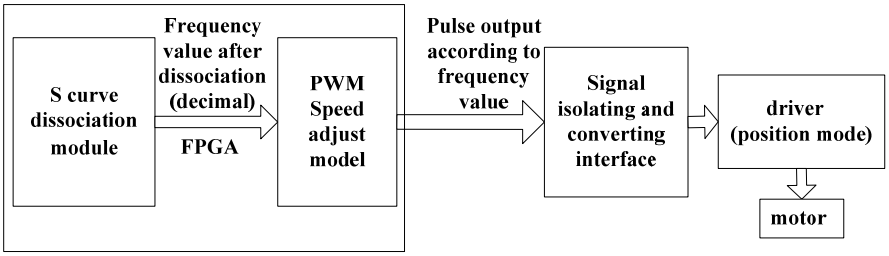


Figure 1. S-curve system structure

Received October 3, 2012; Revised November 3, 2012; Accepted December 6, 2012

The S-curve dissociation module in the Figure 1 extracts several points from continuous speed-time S-curve and converts those speed value to corresponding frequency value (decimal) for every point. The more points extracted, the higher precision of S-curve realized in FPGA.

In this paper the motor drive works in position mode, which changes motor speed by altering the frequency of input signal. Thus, the regulation of motor speed using S-curve controlling algorithm can be achieved by altering the frequency of speed controlling pulse. In Figure 1, In order to adjust the speed of motor, the PWM speed regulation model converts frequency value produced by S-curve module to corresponding frequency pulse output, and different frequency value has different frequency pulse.

2.1 S curve speed dissociation module

The principal of S-curve acceleration and deceleration [5] is shown as Figure 2.

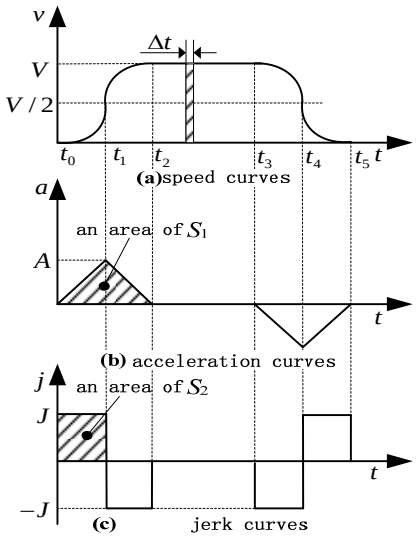


Figure 2. The principle of S-curve acceleration and deceleration

The relationship of speed $v(t)$ and deceleration $a(t)$ is described in formulate (1), and the relationship between acceleration $a(t)$ and jerk $j(t)$ in formulate (2).

$$a(t) = dv(t) / dt \tag{1}$$

$$j(t) = da(t) / dt \tag{2}$$

Without impacting controlling performance, three hypotheses can be drawn:

(a) $t_1 - t_0 = t_2 - t_1 = t_4 - t_3 = t_5 - t_4$

(b) $t_2 - t_0 = T$; $t_1 - t_0 = T / 2$

(c) $v(t_0) = 0$; $v(t_1) = V / 2$; $v(t_2) = V$; $v(t_3) = V$; $v(t_4) = V / 2$; $v(t_5) = 0$

In above formulas, T is accelerating time. Assume that deceleration time equals acceleration time; so the deceleration time is also T . Formulas (3) and (4) are derived from formulas (1) and (2) based on above hypothesis.

While $0 < t < t_1 : v = \frac{1}{2} Jt^2$ (3)

While $t_1 < t < t_2 : v = \frac{1}{4} J(4tT - 2t^2 - T^2)$ (4)

$J = 4V / T^2$, utilizing counter/timer to get running time: $t = \Delta t \times n_c$. This algorithm updates the speed and corresponding output frequency at every cycle Δt, and it is usually microsecond level, n_c is the value of counter. Utilizing formulas (1), (2), (3) and (4), the final speed calculating formulas (5) and (6) can be derived as follows:

While $0 < t < t_1 : v = 2V\Delta t^2 n_c^{\ 2} / T^2$ (5)

While $t_1 < t < t_2 : v = V / T^2 (4T\Delta t n_c T - 2\Delta t^2 n_c^{\ 2} - T^2)$ (6)

The value of v on every point of the S-curve can be derived from formulas (5) and (6). Then decimal frequency value corresponding to different speed can be derived from the formula of speed and frequency $f = kv$ (the unit v of is um/s and f is HZ, k is a constant and it has different value in different system, for experimental system, k is set to 20) [6]. Figure 3 is S-curve's instantiation figure in FPGA.

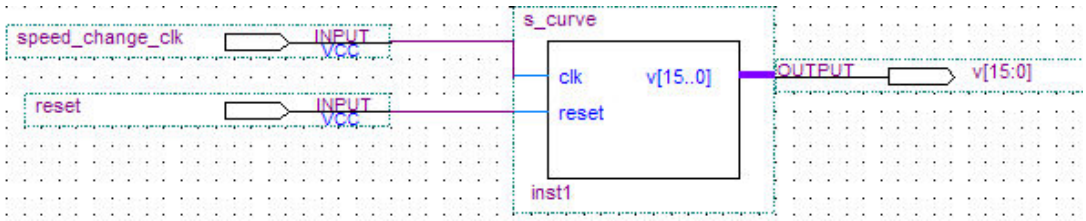


Figure 3. The S-curve model instantiation figure in FPGA

2.2 PWM speed regulation model

Frequency value f derived from corresponding speed value needs to be converted to corresponding pulse output. While the new type of power electronic power components are appearing daily, Pulse Width Modulation (PWM) control ways become mainstream [7], [8] by using all-controlling switch power components. In this paper, motor driver works in position mode and can adjust motor speed by changing the frequency of PWM signal (keeping duty ratio unchanged). Figure 4 is the basic model of PWM.

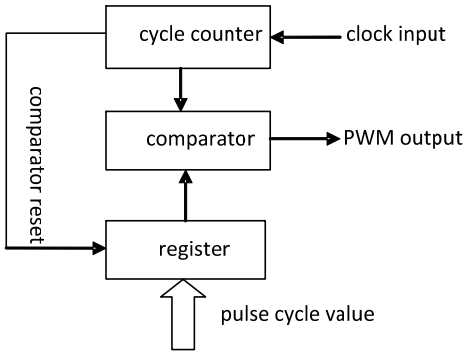


Figure 4. The basic model of pulse width modulation

In Figure 4, circulation counter count extern sampling pulse and its value is added 1 at every sampling cycle. The comparator will compare counter value with value loaded in buffer register. When the output value of counter is less than half of the counting value of pulse cycle, the output of comparator stays low level. When they are equal, the output changes into high level. When the output value is equal to the counting value of pulse cycle, the output level is changed from high to low. Thus a pulse cycle is over. In every cycle of comparator's counting, the comparator outputs different frequency pulse due to different pulse cycles of modulating signal from input, so that the speed regulation of motor can be achieved under the position mode of motor drive.

Figure 5 is the instantiation figure of PWM model. This model designs a common controlled frequency division machine, which can meet various requirement described above using hardware programming language Verilog HDL [9] on Quartus-II [10] researching platform. Compared to Figure 4, reset is reset signal, clk is input clock, cycle is cycle value of pulse (it is the value of f derived from 2.1) and clkout is PWM output signal.

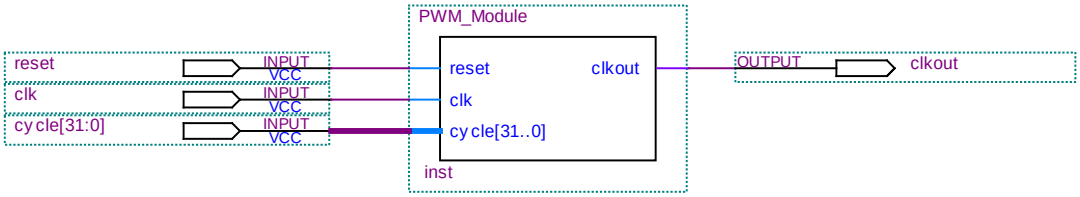


Figure 5. Instantiation figure of PWM model in FPGA

PWM model is realized in FPGA using hardware programming language Verilog HDL and its core code is shown as below.

```
    wire[31:0] duty;                          // occupy proportion
    assign duty = compare_reg/2;               //occupy proportion is always 50%
    reg[31:0] compare_count;//count of pulse
    //regcompare_reg_refresh_flag;
    always @ (negedge reset or posedgeclk)
    begin
    if(!reset)
        begin
        clkout<= 0;
        compare_count<= 0;
        end
    else if (compare_reg != 0)
        begin
    /* when output value of comparator is half of cycle value ofpulse, the output level
of comparator changes to high level.*/
                if(compare_count == duty)
                        begin
                        clkout<= 1;
                        compare_count<= compare_count + 1;
                        end
    /* when output value of comparator is equal to cycle value of pulse, the output level
of comparator changes from high to low. */
                else if (compare_count == compare_reg)
                        begin
                        clkout<= 0;
                        compare_count<= 1;
                        end
                else
                        compare_count<= compare_count + 1;
        end
    else
        begin
        compare_count<= 0;
        clkout<= 0;
        end
    end
```

Timing simulation of PWM model in FPGA is shown as Figure 6.

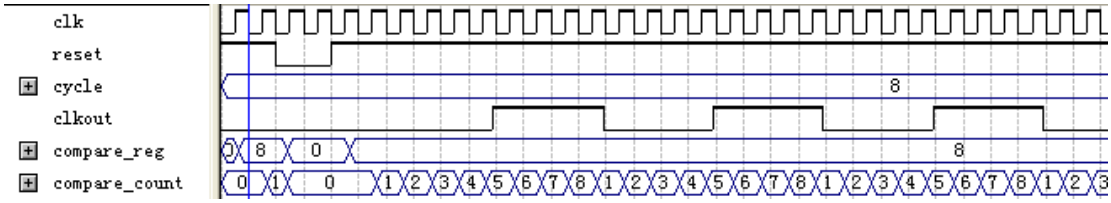


Figure 6. Timing simulation result of PWM model in FPGA

3. Implementation of S curve acceleration and deceleration

In order to simplify operation, acceleration time, constant speed time and deceleration time are set to1 second, speed refresh cycle is set to 200 us. The speed of starting up moment (t=0) is set to zero and stable running stage ($t_2 \le t \le t_3$) is set to 20mm/s. For concrete implementations of each speed stage curve in Figure 2(a), the following corresponding equations can be derived from formulas (5) and (6).

when $0 < t < t_1$ and $t_4 < t < t_5$:

$$v = 16n_c^{\ 2} / 10000(um/s) \tag{7}$$

when $t_1 < t < t_2$, $t_2 < t < t_3$ and $t_3 < t < t_4$:

$$v = 20000 - (10000 - 2n_c)^2 / 2500(um/s) \tag{8}$$

n_c is 50Mhz pulse counter. In the acceleration stage, n_c is set to zero initially and add 1 operation throughout the stage. After 1 second, the acceleration is over and n_c reaches maximum value. In constant speed stage, n_c keeps constant. For the process in speed deceleration stage, the result is completely the opposite of acceleration stage; with n_c decreases as 1 operation is continued to be subtracted from the maximum value. Thus, motor will stop at the end of deceleration stage.

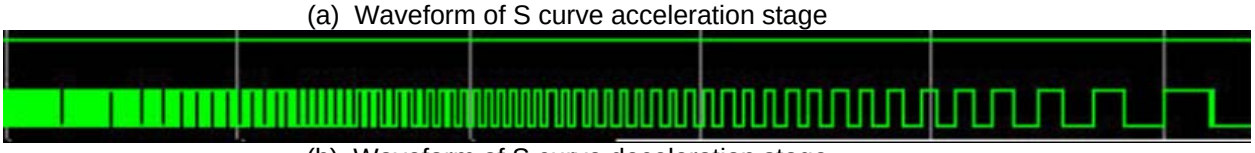
S-curve acceleration and deceleration model is realized in FPGA using hardware programming language Verilog HDL and its core code is shown as below.

```
begin
/*************************************************************************
The first speedup stage
PWM_INITAL_COUNT is the counting value corresponding to minimum initial speed that can
be realized in FPGA, and its precision depends mainly on the control accuracy and speed
of the refresh interval. The higher control precision of the system, the smaller the
refresh interval, the smaller minimum initial speed needed to be realized and this
parameter value is also smaller. When speed value is reduced to a certain degree, its
corresponding frequency pulse output cannot be achieved within the limitation of the
hardware itself. In the test, this value is set to 300. FIR_STAG_TIMES is the counter
value corresponding to the end of the first stage of acceleration and is set to 2500 in
test system.
div_3_result = 16n_c^2 /10000(um/s) is the speed value when 0 < t < t_1. This algorithm
is finally realized by multiplier and divider in FPGA, n_c will add 1 at every rise of
the refresh clock (speed_change_clk), at the end of first acceleration stage n_c =
FIR_STAG_TIMES=2500.
*************************************************************/
```

```
                if (s_count>= PWM_INITAL_COUNT &&s_count<= FIR_STAG_TIMES)
                  begin
                          v <= div_3_result;
                  end
/************************************************************
The second speedup stage
TOTAL_SPEEDUP_TIMES is the counting value corresponding to the end of the second
acceleration     stage     and     is     set     to     5000     in     test     system.
div_1_quo = 20000 – (10000 – 2n_c)^2 / 2500(um / s) is the speed value when t_1 < t < t_2
.This algorithm is finally realized by multiplier and divider in FPGA, n_c will add 1 at
every rise of the refresh clock(speed_change_clk), n_c =TOTAL_SPEEDUP_TIMES=5000 at the
end of second acceleration stage, and will keep its value not changed in constant speed
stage.
************************************************************/
                if(s_count> FIR_STAG_TIMES &&s_count<= TOTAL_SPEEDUP_TIMES )
                       begin
                              v <= 20000 - div_1_quo;
                       end
/************************************************************
The first speeddown stage
The first stage of deceleration is an inverse process of the second stage of
acceleration. FIRST_SPEEDDOWN_STAR_POINT is counting value corresponding to the moment
when    deceleration    stage(the    end    of    the    constant    stage)    is    over,
SECON_SPEEDDOWN_STAR_POINT is the counting value corresponding to the moment when first
deceleration stage(the moment at the beginning of second deceleration) is over.
************************************************************/
if(s_count>=FIRST_SPEEDDOWN_STAR_POINT&&s_count<=SECON_SPEEDDOWN_STAR_POINT)
           begin
                    v <= 20000 - div_1_quo;
           end
/************************************************************
The second speeddown stage
The second deceleration stage is a inverse process of the first acceleration stage.
S_CARVE_END_POINT is the speed value corresponding to the moment when S curve algorithm
is running over
************************************************************/
                if (s_count>SECON_SPEEDDOWN_STAR_POINT &&s_count<= S_CARVE_END_POINT)
                  begin
                          v <= div_3_result;
                  end
```

Combined with PWM speed adjusting model, the simulation result for acceleration and deceleration of the S curve algorithm using Modelsim is shown as Figure 7.

(a) Waveform of S curve acceleration stage



(b) Waveform of S curve deceleration stage

Figure 7. Waveform of S curve acceleration and deceleration stage

Seen from the figure above, the expected change between output pulse frequency in acceleration stage and deceleration stage is consistent.

4. Application experiment

Using motion control platform based on ARM-FPGA[11],[12] to test the effect of S curve algorithm realized in FPGA (EP2C5Q208C8 from Altera), the experimental device using in the test is show as Figure 8.

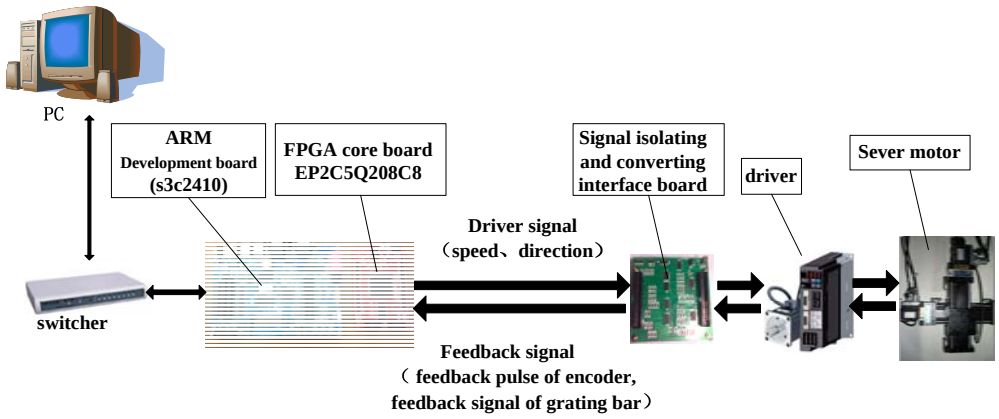


Figure 8. The experimental device in S-curve of acceleration and deceleration

While FPGA is running S-curve algorithm, it is also collecting the data of motor encoder at the same time, and those data will be uploaded to PC through Ethernet to display. The result is shown as figure 9.

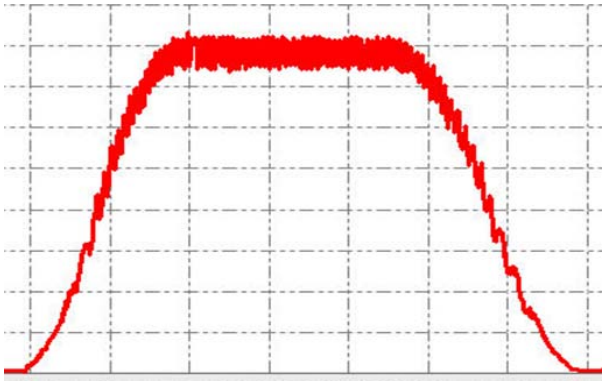


Figure 9. The motion control result of S-curve based on FPGA

Seen from the chart, the curve shape is consistent with the theoretical shape fundamentally. In this testing system, the date collecting cycle for motor encoder is 200us and the collecting date of PC from FPGA has no following process, so the curve is not very smooth. If a display with higher accuracy and more smooth is needed, some other methods can be taken such as reducing the date collecting cycle and using certain data trailing process algorithm.

5. Conclusion

The controller based on hardware programming language has advantages like high system integration and short design cycle, which makes it mainstream in chip designing. For the open motion control system, researching and realizing acceleration and deceleration model using hardware with reusing function and related functional model, taking advantage of

reconfigurable ability (algorithm reconfigurable) of the programmable logic devices FPGA, then the motion control chip whose functions are customized can be achieved flexibly according to requirement.

Acknowledgements

This work has been supported by Open Research Foundation of State Key Lab. of Digital Manufacturing Equipment and Technology, HUST, China (Grant No.DMETKF2009017).

References

[1] Chen Zeyu, Zhao Guangyao. Fuzzy Control Strategy and Simulation for Dual Electric Tracked Vehicle Motion Control. *Applied Mechanics and Materials*. 2011; 130-134: 309-312.
[2] Tan KK, Huang SN, Dou HF, Lee TH, Chin SJ, Lim SY. Adaptive robust motion control for precise trajectory tracking applications. *ISA Transactions*. 2001; 40(1); 57-71.
[3] Zhang Huayu, Wang Fumao, Xu Fangquan. Study on the Algorithm of S Curve Acceleration/deceleration. *Modern Manufacturing Technology and Equipment*. 2006; 2: 21-22.
[4] Yang Yan, Wang Yunkuan, Song Yinghua. Implementation of Acceleration/Deceleration of NC System Based on FPGA. *Manufacturing Technology & Machine Tool*. 2007; 2: 31-34.
[5] Lin Yisong, Tang Zhaohong, Ou Ruixiang, Lin Jinping, Gong Deming. An Ac/decelaration Calculation Method for S Curve. *Manufacturing Technology & Machine Tool*. 2005; 11:74-75.
[6] Pan Wu, Research and Design an Reconfiguration Motion Control System Based on Networks. *Control System Based on Networks*. 2009; 5:59-60.
[7] Kumar, Mallisetti Rajesh, Lenine, Duraisamy Babu, Ch Sai. A variable switching frequency with boost power factor correction converter. *Telkomnika*. 2011; 9(1): 47-54.
[8] Li Zhaohui, Shen Gang. Design of digital pulse width modulator based on CPLD. *Electrical Drive Automation*. 2004; 6(3):7-9.
[9] Sutikno, Tole. FPGA for robotic applications: From android/humanoid robots to artificial men. *Telkomnika*. 2011; 9(3): 401-402.
[10] William Kleitz. Digital Electronics with VHDL (Quartus II Version). Publisher: Prentice Hall. Copyright: 2006.
[11] Yang Guangyou, Li Ming, Zhang Daode and Xu Wan. Ethernet Interface of High Speed Data Acquisition System Based on ARM-FPGA. *Advanced Science Letters*. 2011; 4: 2538–2542.
[12] Yang Guangyou, Li Ming, Zhang Shuangqing and Xu Wan. Research and implementations of the IEEE 1588 Precision Time Protocol based on ARM-Linux. *Advanced Materials Research.* 2011; 156-157: 1492-1496.