Design of Ion Implantation and Deposition Signal Generator Based on FPGA and Touch Screen

Weiquan Shu* and Yanhua Zheng
Zhejiang International Maritime College, Zhoushan, Zhejiang, China

*Corresponding author email: swqyx@foxmail.com

Abstract. In order to improve the operation precision of ion implantation and deposition signal generator, to fix its arithmetic error and reach 1μs as the pulse adjustment unit, use FPGA and touch screen technology to improve the performance of signal generator. On the one hand, the FPGA board can directly affect the FPGA chip through Verilog to reduce the program operation time, on the other hand, replacing with crystal oscillator with higher frequency can make the program run faster and minimize the arithmetic error. The combination of FPGA and crystal oscillator can easily realize the clock input of MHz level and output pulse with 1μs as the adjustment unit, which has already been applied to signal generator as signal output source.

Keywords: Touch screen; FPGA; Crystal oscillator; Clock division.

1. Background and Significance
With the rapid development of modern science and technology, the requirements of metal surface properties (anti-wear, anti-corrosion, anti-fatigue, etc.) are increasing day by day, which makes the ion implantation technology come into being. Ion implantation and deposition technology is an important technology in the welding surface modification research area. It can realize the vertical and uniform ion implantation and deposition on the surface of complex shaped parts, and has wide application prospects in the field of surface modification.

Ion implantation and deposition signal generator is aimed at ion implantation technology, by adjusting four input factors (main arc pulse width, the bias arc pulse width, frequency and the delay time), to control the whole injection and deposition process, and ensure the integrity and efficiency of ion implantation and deposition process.

FPGA (Field-Programmable Gate Array) is a product of further development based on PAL, GAL, CPLD and other programmable devices. It appears as a semi-customized circuit in the field of Application Specific Integrated Circuit (ASIC). It not only solves the shortcomings of customized circuit, but also overcomes the disadvantage of the limited gate number of original programmable devices. It is written by hardware description language (Verilog or VHDL) and can be quickly burned to FPGA for testing after simple synthesis and layout, which is the main stream of modern IC design verification technology. FPGA can use programmable control hardware to execute some instructions or increase the frequency of global clock to speed up the operation of the program, it has higher accuracy compared with PLC by minimizing program operation time (even can be ignored) and has no effect on parameter settings.
2. Technical Analysis of FPGA

2.1. FPGA Chip Model and Parameters
The FPGA chip adopted this time is ACEX series product of Altera, model: EP1K30TC144-3, it has 144 pins and 102 I/O ports. The program language is Verilog, 1μs as the pulse adjustment unit can be realized with 1MHz removable external crystal oscillator MCO-1510A as clock input. The program operation time can be greatly reduced if replacing with higher frequency crystal oscillator, and the clock frequency needed by pulse width adjustment can be obtained by frequency division.

2.2. FPGA Programming Environment
The ACEX series FPGA need to be programed under Quartus II 8.0. The main design process includes: create projects for specific models; input design files with Text Editor or Block Editor (use Text Editor if programming language is Verilog); compile and detect errors; assign pin; load configurations and simulate.

2.3. The Introduction of Development Board
The required FPGA chip has been embedded into a development board (as shown in Figure 1). It supports DB25 parallel port which can be connected to PC. The program download mode is JTAG; the crystal oscillator is removable and can be replaced with different frequency according to the actual needs; 32 I/O ports as output and two rows of extended I/O sockets are also added. The overall design scheme and configuration can meet the requirements.

3. The Design of FPGA Program
This chapter mainly introduces the design of FPGA program. On one hand, FPGA program uses hardware description language Verilog to reduce the program operation time, on the other hand, replacing with crystal oscillator with higher frequency can make the program run faster and minimize the arithmetic error. In addition, the precision of pulse width adjustment by PLC is 5μs, and with the combination of FPGA and crystal oscillator, the clock input of MHz level can be easily realized, and output pulse of 1μs as adjustment unit.

3.1. FPGA Main Program and Analysis
The design concept of this program is: use values of “widtha”, “delayb” and “widthc” correspond to the main arc pulse, delay time and bias pulse respectively, when the calculated value is the same as the set value (“mainw”, “delay”, “biasw”), the “pwm” output takes corresponding changes (“pwm1” is effective at high level, “pwm2” is effective at low level), takes one “period” as the period and loops, thus produces the continuous pulses.

The general process of program design is as follows:
First, build a module, define the input, output and the names of other counters and registers; define the bit length of the counters and registers according to the required parameter setting range. For example,
[14:0] means the bit length of the register is 15, so the value range is 0-2^{15}, can realize two pulses with frequency below 20Hz. (The period is longer than 50ms).

Secondly, write the loop part, use “always” statement as loop, and trigger a statement every time a rising edge of the clock is reached.

Reset control:
IF (!rst_n)
BEGIN
  widtha = 15'b0;
  delayb = 10'b0;
  widthc = 15'b0;
  pwm1 = 0;
  pwm2 = 1;
END

In the condition part, “rst_n” is the reset signal, low level is effective. At low level, the counters corresponding to the main arc pulse width, bias pulse width, delay time and the two pulse output signals will be reset to the initial state.

Main arc pulse width control:
IF (widtha < period)
BEGIN
  widtha = widtha + 1;
  IF (widtha <= mainw)
  BEGIN
    pwm1 = 1;
  END
  ELSE
  IF (widtha > mainw & & widtha < period)
  BEGIN
    pwm1 = 0;
  END
END

Activate the condition statement when the value of the main arc pulse counter is less than the value of period. “widtha” will add 1 each rising edge of clock. When the value of “widtha” is less than the set value of the pulse width, the main arc signal outputs the high level. When the value of “widtha” is between the set values of the pulse width and the period, the main arc signal outputs the low level.

Bias pulse width control: it is almost as same as the main arc pulse width control in principle, the difference is that the launch of bias pulse needs to be triggered after a certain delay time. Before triggering, the bias pulse “pwm2” is high level, and after triggering, the pulse width is set to "0" as the output of negative voltage.

Delay time control:
delayb = delayb + 1;

……
IF (delayb <= delay)
BEGIN
  ……
END
ELSE
IF (delayb > delay)
BEGIN
  ……
END
ELSE
IF (delayb == 1024)
BEGIN

……
delayb = 10'b1111111111;
END

The parameter “delayb” is used as the delay time counter. When it is less than or equal to the pre-set value, the bias pulse signal outputs high level. After a certain delay time, it starts to execute the control and loop statement of bias pulse, and sets the maximum value of counter to 1024 to prevent program error. Finally, the module ends, and the program exits.

3.2. UART Communication Protocol and Analysis

3.2.1. Introduction of UART. UART (Universal Asynchronous Receiver/Transmitter) is a common serial data communication bus protocol, which can realize full duplex data receiving and sending, and is often used for communication between FPGA and peripheral devices. In order to realize the communication between FPGA and touch screen, and achieve a touch effect, UART communication protocol must be designed.

UART adopts asynchronous communication mode, which is generally divided into sender and receiver (as shown in Figure 2). The receiver is used to receive the serial data sent by another UART and store it in the data register, waiting for the data bus to get; the sender is used to convert the data in the data register into serial data and send it out. The two sides have their own separate clocks, and the frequency of data transmission needs to be agreed by both sides, that is to say, in the process of communication between FPGA and touch screen, the sender (TXD) of touch screen and the receiver (RXD) of FPGA have their own clocks. When they communicate, they should have the same baud rate (the number of carrier modulation state changes per unit time), otherwise, data transmission error will be generated, resulting in failing getting data normally.

3.2.2. UART communication protocol design. Because FPGA only needs to receive and read data from the touch screen and does not need to feed back as a sender, it only needs to write the receiver protocol. The global clock frequency of FPGA is 1 MHz. Considering the baud rate of the touch screen is 9600 bauds, in order to make the baud rate of the receiver the same as it, it is necessary to divide the clock frequency of 1 MHz, the frequency division number is $\frac{1000000}{9600} = 104$. In the receiver, the start bit takes the low level as the start signal of data reception, when a set of data is sent, if the stop bit detects the high level, it is ready to receive the next set of data.

The overall structure of the program is divided into three parts: definition part, control part and end part.

The definition part is to define the module, shift register, data register and counter. In this program, two 8-bit shift registers, “data_in_1” and “data_in_2” are used as low 8-bit and high 8-bit respectively. A 16-bit data register is used to store and output the combination of low and high data. A state register is used to display status in the case statement. A counter is used to divide the frequency of global clock, one for global clock, one for reset switch and one for serial data input.

The control part uses control statements to receive data and store it in the shift register, and uses combination of a series of conditional statements and loop statement to transfer the data in the shift register to the data register for output. In this program, each rising edge of the clock triggers the "always" loop statement, and starts to receive data and store it when the reset switch is high level. The case statement is used to execute corresponding commands when the counter reaches a state. These commands mainly include the identification of the start bit, depositing of data, and identification of the stop bit.

The end part is to end modules, loop statements, conditional statements, etc., usually using “end” and “endmodule” as the end command.

There are four main technical points in the program: falling edge detection, data intermediate sampling, clock frequency division, 8-bit data transferring to 16 bit.

Falling edge detection: the purpose is to make the start bit of the receiver can start to work normally, as the “0” state in the program.

0 : BEGIN
now <= rxd;
pre <= now;
IF (pre & & !now)
    state <= 1;
ELSE
    state <= 0;
END

“now” is the current received bit sampled by sampling clock, “pre” is the previous received bit, that is, when “pre” is 1 and “now” is 0, it is the falling edge, and the logical relationship is defined as “pre&&!now”. It must be low level during the sampling period after the falling edge, which can be used as the start bit of the receiver, as the 1 state in the program.

Clock frequency division: in order to synchronize the clock of sender and receiver, it is necessary to divide the 1MHz global clock frequency. The clock of this program is set to 9600 bauds, and the clock of sender can be automatically adjusted to 9600 bauds through the touch screen, while the receiver needs to calculate the frequency division number, that is, 1000000 / 9600 = 104, clock with a period of 1μs counts 104 times, which is the sampling period.

Data intermediate sampling: because the clock of the receiver and the sender is not fully synchronized, the clock deviation will bring errors to the reception. To solve this problem, the sampling time can be placed in the middle of each sampling bit.

1 : IF (count == 52)
BEGIN
    IF (rxd == 0)
        state <= 2;
    ELSE
        state <= 0;
    count <= 0;
END
ELSE
BEGIN
    state <= 1;
    count <= count + 1;
END

After the falling edge detected, when the counter reaches baud/2, the start bit starts to sample, then the shift register samples every baud rate, so that each sampling point is at the middle of two sampling bits. In the program, the global clock is 1 MHz, the baud rate is 9600, so counting up to 104 is the period of the receiver clock, 52 is the intermediate point.

8-bit data transferred to 16 bit: since the setting range of the main arc pulse width in the program is up to 5000μs, and the period range is 20-200Hz (5000-50000μs), which means only 16-bit data transmission can be satisfied, while one frame in UART is composed of 8 bit by default, so it is necessary to set the high 8-bit and the low 8-bit, and combine them to 16-bit data. The so-called low 8 bit is the last 8 bit of 16-bit data, and the high 8 bit is the first 8 bit of 16-bit data. In the program design, the data of the low 8 bit is received, and after the stop bit detected, the start bit of the high 8 bit starts to work.

3.3. Simulation Result

In the simulation result of UART receiver, “rxd” is assumed to be the analog wave input with the period of 208μs and duty cycle of 50%. Start from the first falling edge, the start bit samples the low level. After one sampling period (104μs), it is "1", and then after another, it is "0", and so on. The received data of low 8 bit is 10101010. After that, the stop bit samples the next “rxd” which is high level, the start bit of the high 8 bit starts to receive high 8-bit data, the specific result is as same as the low 8 bit, which is 10101010. Finally, by combining the high 8 bit with the low 8 bit, the 16-bit data is transferred from the two 8-bit shift registers to the 16-bit data register [15:0] data, outputs 16-bit data: 0101010101010101. Through the simulation, it can be seen that the pulse output part is feasible,
communication protocol module works fine, the final result is qualified. Table 1 shows the waveform output adjustment range.

Table 1. Parameter adjustment ranges of signal generator in FPGA programming

| waveform       | main arc pulse width | delay time | frequency | high voltage pulse width |
|----------------|----------------------|------------|-----------|--------------------------|
| square wave    | 0-5 ms               | 0-1000 μs  | 20-500 Hz | 0-200 μs                 |

4. Conclusion
The advantages of FPGA are as follows: 1. less arithmetic error: the FPGA development board can directly affect the FPGA chip through Verilog to reduce the program operation time, and replacing with crystal oscillator with higher frequency can make the program run faster and minimize the arithmetic error. 2. On the development board, the removable crystal oscillator is used as the clock input which can reach the frequency of dozens MHz, after frequency division, the 1 MHz clock needed by pulse width control can be obtained. Therefore, 1μs as the adjustment unit can be designed by using the counting principle, which is better than PLC.
The purpose of researching and designing FPGA program is to solve the problem of arithmetic error caused by the program operation in the software of PLC, and to realize 1μs as pulse width adjustment unit, so as to achieve better effect.

References
[1] Yongjin Chen, Qingnan Shi, Xiaqing Zuo, et al. 2003. Improvement of Surface Property of Metallic Materials—the Development and Application of Ion Implantation [J]. Surface Technology 2003 (06): pp 4-7.
[2] Weijian Hao. 2013. Pulse Width Modulation Signal Generator Based on FPGA [J]. Computer Engineering 2013 (02): pp 260-264.
[3] Lianzhen Huang, Jiangang Li, Dongjun Zhang, 2012. Frequency-controllable Sine Signal based on PWM and Its Implementation on FPGA[J]. Frontiers of Mechanical Engineering 2012, p 73.
[4] Zhe HU, Jun ZHANG, Xiling LUO. A Novel Design of Efficient Multi-channel UART Controller Based on FPGA[J]. Chinese Journal of Aeronautics 2007, 01: pp 66-74.
[5] Jie Liu, Minggang Wang, Weifeng Yang, et al. 2010. Research of Plasma Immersion Ion Implantation Technology and Equipment [J]. Semiconductor Technology 2010 (07): pp 626-629.
[6] Naouar M.W, Hania B.B, Slama-Belkhodja I, et al. 2013. FPGA-based Sliding Mode Direct Control of Single Phase PWM Boost Rectifier[J]. Mathematics and Computers in Simulation 2013, p 91.
[7] Xiaofei Zhang, Ganggang Qin, Yang Yang. 2012. Introduction to FPGA technology and typical project development examples [M]. Beijing: Chemical Industry Press 2012.
[8] P.H. Rose, Concepts and designs of ion implantation equipment for semiconductor processing Review of Scientific Instruments 77, 111101 (2006).
[9] Paul K. Chu, Chung Chan. Applications of plasma immersion ion implantation in microelectronics — a brief review[J]. Surface & Coatings Technology. 2001 (1).
[10] G.A. Collins, R. Hutchings, K.T. Short, J. Tendys, X. Li, M. Samandi. Nitriding of austenitic stainless steel by plasma immersion ion implantation[J]. Surface & Coatings Technology. 1995.
[11] Bing Liu, Danyin Zou, Lei Feng et al. An FPGA-Based CNN Accelerator Integrating Depthwise Separable Convolution[J] Electronics, 2019, 8(3).
[12] Subhranjit Mitra, Paramita Chattopadhyay Design and implementation of flexible Numerical Overcurrent Relay on FPGA[J] International Journal of Electrical Power and Energy Systems, 2019, 104.