FPGA Implementation of SPI Bus Communication Based on State Machine Method

Jiayi Qiang[1], Yong Gu[2] and Guochu Chen[1*]

[1]College of Electrical Engineering, Shanghai Dianji University, Shanghai, 201306

[2]College of Electrical Engineering, Shanghai Dianji University, Shanghai, 201306

[*]Corresponding author's e-mail: chengc@sdju.edu.cn

Abstract. The SPI bus is a synchronous serial interface data bus with full duplex, few signal lines, simple protocol, and fast transmission speed. Based on these characteristics, parallel high-speed computing with FPGA is used to meet device expansion and experiment in high-rate environments. This paper introduces the structure and working principle of SPI communication bus, analyzes its timing structure and four working modes, and uses this state machine method to realize its SPI bus communication function on FPGA. The module circuit of SPI is written by Verilog hardware description language, and the waveform is simulated in vivado simulator. After the simulation waveform analysis, the feasibility of the state machine method is verified.

1. Introduction

SPI, the Serial Peripheral Interface, is a synchronous serial interface technology introduced by Motorola. The SPI bus is physically implemented by a module called a Synchronous Serial Port on a microprocessor control unit (MCU) connected to a peripheral microcontroller (PIC micro) [1], which allows MCU performs high-speed data communication with various peripheral devices in full-duplex synchronous serial mode.

Among many serial buses, SPI mid-line has great advantages compared with other common buses such as I2C bus, CAN bus, USB, etc. For example, the data transmission speed of SPI bus can reach several Mbps, which is faster than other serial buses obviously [2-3]. SPI is mainly used in EEPROM, Flash, Real Time Clock (RTC), Digital to Analog Converter (ADC), Digital Signal Processor (DSP) and Digital Signal Decoder [4].

In this paper, we introduce the structure and working principle of SPI communication bus, analyze its timing structure and four working modes, and uses this state machine method to realize its SPI bus communication function on FPGA. The module circuit of SPI is written by Verilog hardware description language, and the waveform is simulated in vivado simulator.

2. SPI protocol

2.1. SPI bus structure

In the master-slave mode, the SPI specifies that the communication between the two SPI devices must be controlled by the master. A master device can control multiple slave devices by providing a clock and Slave Select for the slave device. The SPI protocol also stipulates that the slave device's clock is

provided by the master device to the slave device through the SCK pin. The slave device itself cannot be generated. Or control the clock, the Slave device does not work without the clock [5].

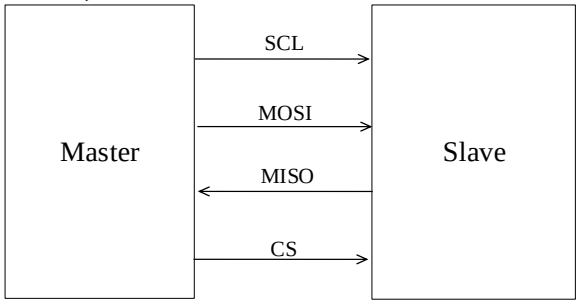

Figure 1. SPI bus structure block diagram

The master and the slave are connected by four signal lines, which are SCK, MOSI, MISO, CS, and their definitions are as follows: SCK: Synchronous clock signal, used to synchronize data transmission between master and slave. The host controls the output [6]. The slave receives and transmits data on the edge of SCK; MOSI: host output, slave input signal, host is on the rising edge (or falling edge) sends data to the slave through the signal line, and the slave receives the data through the signal line on the falling edge (or rising edge); MISO: master input, slave output signal, slave on rising edge (or The falling edge transmits data to the host through the signal line, and the host receives the data through the signal line on the falling edge (or rising edge); CS: the slave chip selects the signal, and the master controls the output.

2.2. SPI bus working principle

When there is no data to be transferred between the master and the slave, the host controls the SCK output idle level, the CS output is inactive, and the SPI bus is idle; when there is data to be transmitted, the host controls the CS output active level, SCK The clock signal is output and the SPI bus is in operation; on a certain clock edge, the master and the slave simultaneously transmit data and transmit the data to MOSI and MISO respectively; on the next clock edge, the master and the slave receive data at the same time, respectively, MISO And the data on MOSI is received and stored[7]; when the data is completely transmitted, the host controls the SCK output idle level, the CS output is inactive, and the SPI bus returns to the idle state. At this point, a complete SPI bus data transfer process is completed.

The SPI bus has the following main features:

Single Master Multi Slave is supported by Master-Slave control. The SPI specifies that communication between two SPI devices must be controlled by the master to slave devices. That is to say, if the FPGA is a host, whether the FPGA sends data to the chip or receives data from the chip, the chip select signal CS and the serial clock signal SCK must be generated by the FPGA when writing the Verilog logic. At the same time, a master can set multiple Chip Select to control multiple slaves.

The SPI bus also transmits the clock signal while transmitting data, so the SPI protocol is a synchronous (Synchronous) transmission protocol. The Master generates a clock signal based on the data to be exchanged to form a clock signal. The clock signal controls the timing of the two SPI devices to exchange data and when to sample the received data through clock polarity (CPOL) and clock phase (CPHA). Ensure that data is transmitted synchronously between the two devices.

2.3. SPI Clock Specifications

The SPI bus transmission has a total of four modes, which are defined by clock polarity (CPOL, Clock Polarity) and clock phase (CPHA, Clock Phase), where the CPOL parameter specifies the level of the SCK clock signal idle state. CPHA specifies whether data is sampled on the rising edge of the SCK clock or on the falling edge. The timing diagram for these four modes is shown below:

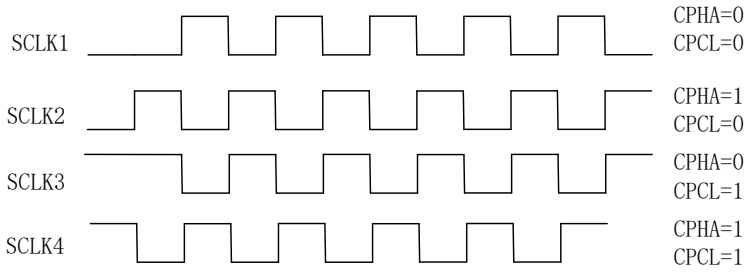

Figure 2. Four working mode timing diagram

Mode 0: CPOL = 0, CPHA=0. The SCK serial clock line idle is low, data is sampled on the rising edge of the SCK clock, and data is switched on the falling edge of the SCK clock;

Mode 1: CPOL = 0, CPHA=1. The SCK serial clock line idle is low, data is sampled on the falling edge of the SCK clock, and data is switched on the rising edge of the SCK clock;

Mode 2: CPOL = 1, CPHA = 0. The SCK serial clock line idle is high, data is sampled on the falling edge of the SCK clock, and data is switched on the rising edge of the SCK clock;

Mode 3: CPOL = 1, CPHA = 1. The SCK serial clock line idle is high, data is sampled on the rising edge of the SCK clock, and data is switched on the falling edge of the SCK clock.

3. Software design of SPI communication bus interface

The SPI master module designed in this paper mainly accomplish the following work:

(1) Converting 8-bit parallel data received by the host into serial data and transmitting it to the slave.

(2) Receive serial data from the slave, convert it to parallel data, and output it through the parallel port.

(3) Output the input signal required by the slave, the clock signal SCK, and the chip select signal CS.

3.1. SPI main module design

In the process of data string conversion, the register must be used to store temporary data. In general, one transmit register is required to transmit data, and one receive register is required to receive data.

The SPI module written by Verilog includes clock, reset, enable, parallel input and output, and completion flags in addition to the four lines for SPI communication. The block diagram is as follows:

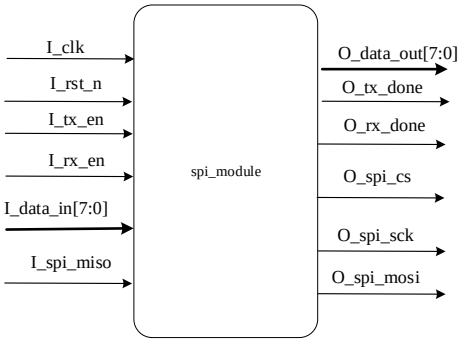

Figure 3. SPI bus protocol model block diagram

3.2. State machine design

The easiest way to achieve the timing of mode 0 above is to design a state machine. The above SPI bus communication process is converted into a state machine step. The following figure shows the timing diagram of mode 0.

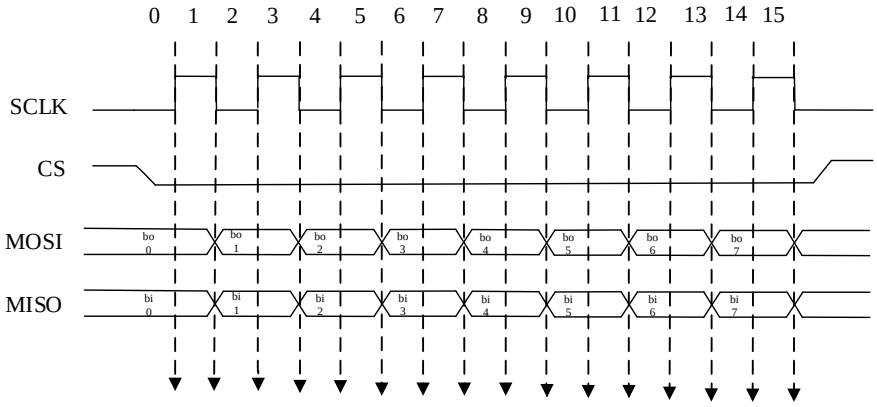

Figure4. Timing diagram of mode 0.

Send: When the FPGA sends a byte (8-bit) data to the slave through the SPI bus, the FPGA first sets the CS/SS chip select signal to 0, indicating that it is ready to start transmitting data. The whole process of sending data can be divided into 16 states:

- State 0: SCK is 0, MOSI is the highest bit of data to be transmitted, I_data_in [7];
- State 1: SCK is 1, MOSI remains unchanged;
- State 2: SCK is 0, MOSI is the next highest bit of data to be transmitted, I_data_in [6];
- State 3: SCK is 1, MOSI remains unchanged;
- State 4: SCK is 0, MOSI is the next bit of data to be sent, I_data_in [5];
- State 5: SCK is 1, MOSI remains unchanged;
- State 6: SCK is 0, MOSI is the next bit of data to be sent, I_data_in [4];
- State 7: SCK is 1, MOSI remains unchanged;
- State 8: SCK is 0, MOSI is the next bit of data to be sent, I_data_in [3];
- State 9: SCK is 1, MOSI remains unchanged;
- State 10: SCK is 0, MOSI is the next bit of data to be sent, I_data_in [2];
- State 11: SCK is 1, MOSI remains unchanged;
- State 12: SCK is 0, MOSI is the next bit of data to be sent, I_data_in [1];
- State 13: SCK is 1, MOSI remains unchanged;
- State 14: SCK is 0, MOSI is the lowest bit of the data to be transmitted, I_data_in [0];
- State 15: SCK is 1, and MOSI remains unchanged.

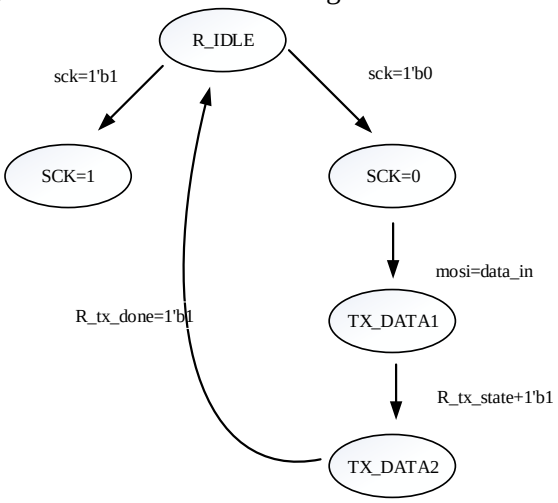

Figure 5. Send process state machine block diagram

The figure above shows the state machine block diagram of the transmission process. After a byte data is transmitted, a transmission completion flag O_tx_done is generated and the CS/SS signal is pulled high to complete a transmission. By observing the above state, it can be found that the state in which the state number is odd is actually the same, so when the code is written, in order to streamline the code, the state with the odd state number can be integrated.

Receive: When the FPGA receives one byte (8-bit) of data from the slave through the SPI bus, the FPGA first sets the CS/SS chip select signal to 0, indicating that it is ready to start receiving data. The whole process of receiving data can also be divided. It is 16 states, but unlike the sending process, in order to ensure that the received data is accurate, it must be sampled in the middle of the data.

After receiving one byte of data, a reception completion flag O_rx_done is generated and the CS/SS signal is pulled high to complete the reception of the data. By observing the above state, it can be found that the state in which the state number is even is actually the same, so when the code is written, in order to streamline the code, the state with the even number of states can be integrated. And this is just the opposite of the state of the sending process.

4. Design simulation, synthesis and implementation

This design was simulated on the vivado Simulator software platform and verified in the actual project. In the actual test, the system is running well, stable and reliable. Figure 9 is a waveform of a successful data transmission in an experiment.

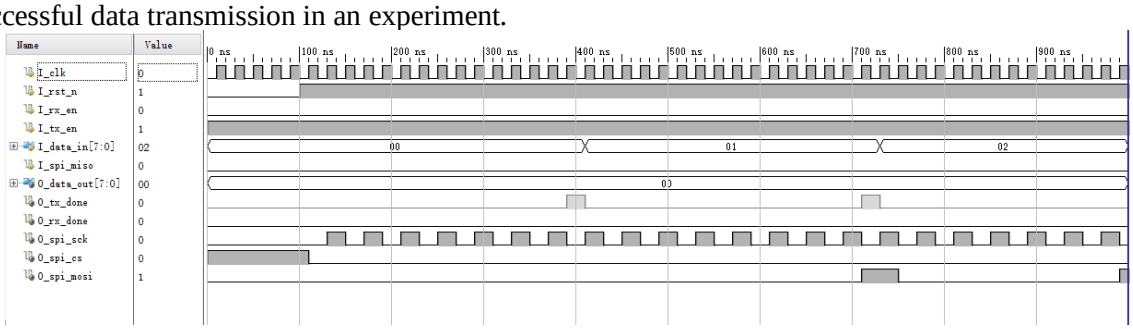

Figure 6. SPI module simulation waveform

It can be seen from the figure that the idle level of sck is low level and the time of receiving data is rising edge, so the working mode of the SPI module is CPOL=0, CPHA=0, which is consistent with the previous design. When the transmit enable terminal tx_en=1, CS=1, the bus is in an idle state. When CS=0, sck starts to output a clock signal, and the bus starts to transmit data. On the rising edge of the first cycle, the host sends the highest bit of data_in to the slave by mosi, and sends tx_done=1 when transmitting to the 8th bit, indicating that the 8-bit data transfer is complete. In summary, the function of the SPI module is correct.

5. Conclusion

SPI has full-duplex, low signal lines, simple protocol, fast transmission speed, etc., which makes it widely used in the field of data transmission. This simple and easy-to-use feature, more and more chips integrate this communication. Agreement. This paper first introduces the communication protocol of SPI, designs a SPI host module conforming to the SPI bus specification with Verilog hardware description language, and performs functional simulation. The SPI host module has the correct function, stable operation and strong scalability. Due to the wide range of applications of the SPI bus and the advantages of FPGA reconfigurability, the module can be easily extended with hardware as needed to increase its functionality.

References

[1] F. Leens, (2009) An Introduction to I2C and SPI Protocols. In: IEEE Instrumentation & Measurement Magazine. Beijing. pp. 8-13.

[2] S. Wang, (2010) Design and Implementation of Serial Peripheral Interface SPI Based on FPGA. Control & Automation, pp.117-119.
[3] W.L.Li, D.P. Y. (2007) Implementation of Asynchronous Serial Port and Synchronous Serial Port Conversion Using FPGA. Electronic Engineer, pp.52-53.
[4] S.Zhang, W.Li. (2014) Modular design method of FPGA. Journal of Electronic Measurement and Instrument, pp. 560-565.
[5] F.J.Sun, C.X.Yu.(2005) Verilog Implementation of SPI Serial Bus Interface. Modern Electronic Technique, pp. 105-106,109.
[6] W.Mai, W.Liu. (2007) Design and Implementation of SPI Interface Based on FPGA and MSP430. Instrumentation Users, pp. 100-102.
[7] W.C.Zhu, S. ,Zhang, H.L.Jiang. (2017) Design of high speed data communication interface based on ARM and FPGA. Journal of Guilin University of Electronic Technology, pp. 293-297.