A Configurable SPI Interface Based on APB Bus

Jiang Yang[1], Yile Xiao[2], Dejian Li[2], Zheng Li[2], Zhijie Chen[1], Peiyuan Wan[1*]
[1]Beijing Embedded System Key Lab, Faculty of Information Technology, Beijing University of Technology, Beijing, China
[2]Beijing Smartchip Microelectronics Technology Company Limited, Beijing, China
wanpy@bjut.edu.cn

***Abstract*—A configurable SPI interface based on APB bus is proposed in this paper. The SPI interface transmission information is configured through APB bus, which can realize the master-slave mode switching, 4 kinds of clock transmission mode switching and MSB/LSB communication mode switching. Meanwhile the transmission rate can be selected and the transmission length can be variable from 1 to 32 bits. Based on the SPI transmission protocol, the proposed design uses a finite state machine method to control the transmission timing of the SPI interface. The design was verified through RTL simulation and FPGA verification. The verification results show that the functional of SPI interface is correct and the data transmission is stable.**

Keywords—SPI interface; APB bus; Configurable; FPGA

I. INTRODUCTION

SPI bus is a synchronous serial interface technology introduced by Motorola[1]. It allows the CPU to communicate and exchange data with various peripheral devices in a serial transmission method[2]. It is a high-speed, full-duplex communication bus, which means that data can be sent while receiving data. Compared with the IIC bus, the transmission speed is faster. Because of its obvious advantages such as easy-to-use and fast communication speed, more and more chips integrate SPI communication protocol. Which has been widely used in embedded systems, such as DSP Flash expansion scheme based on SPI interface[3].

SPI protocol stipulates that communication needs to be carried out between SPI Master and SPI Slave, and four transmission modes are agreed upon. However, the functions of the existing SPI design are not comprehensive enough. Some SPI interfaces have a single function and only have one of the master/slave functions; some SPI interfaces only support one or two clock transmission modes; Some SPI interfaces do not support four-wire working mode and lack versatility; and the transmission rate and transmission data length of some SPI interfaces cannot be flexibly configured, and the application scenarios are single. Therefore, in order to provide maximum flexibility for SOC design and FPGA development, the configurable SPI interface based on APB interface proposed in this paper solves the above problems. The SPI interface can be configured as Master/Slave mode. The communication rate of the SPI interface can be set[4]. The data transmission length of the SPI interface can be changed. The transmission direction MSB/LSB of the SPI interface can be set and 4 kinds of clock transmission modes can be set. And the data transmission in these different situations is realized through the configuration of the relevant control register through the APB interface[5]. The SPI interface designed in this paper is highly feasible, convenient and flexible. It can meet the different needs of users, has broad application scenarios, and has strong market competitiveness.

This paper first introduces SPI transmission protocol, then divides the internal structure of SPI module in detail according to the protocol. Immediately afterwards, this paper designs the registers and SPI state machine of the interface module in detail. Finally, through RTL simulation and FPGA verification, the design can achieve SPI communication correctly.

II. DESIGN SCHEME

A. SPI Transmission Process

SPI has four signal lines: MOSI, master data output, slave data input; MISO master data input, slave data output; SCK clock signal, generated by the master; CS chip select signal, generated by the master, when low level CS is valid[6]. In addition, SCK clock polarity (CPOL) and phase (CPHA) can be configured as required. CPOL=1 means SCK high level is valid; CPOL=0 means SCK low level is valid. The clock phase signal CPHA=0 indicates that the data is valid on the first SCK edge after the signal CS is declared; And CPHA=1 indicates that the data is valid on the second SCK edge after the signal CS is declared. The SPI master and slave clock phase and polarity settings must be the same. The four data transmission timings of the SPI protocol are shown in the Fig. 1.

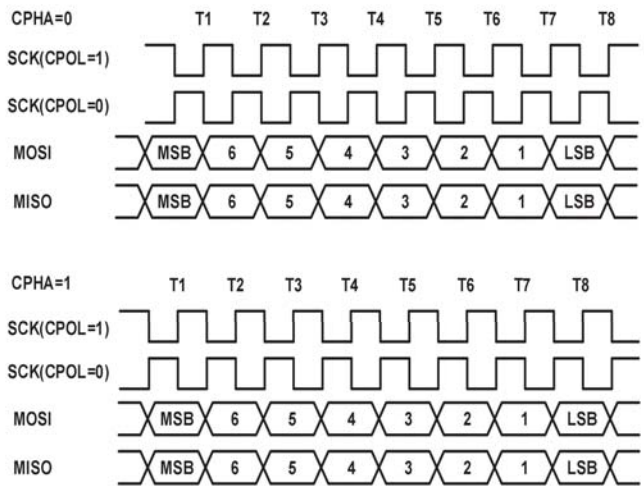

Figure 1. SPI transmission timing

978-1-7281-6879-1/20/$31.00 ©2020 IEEE

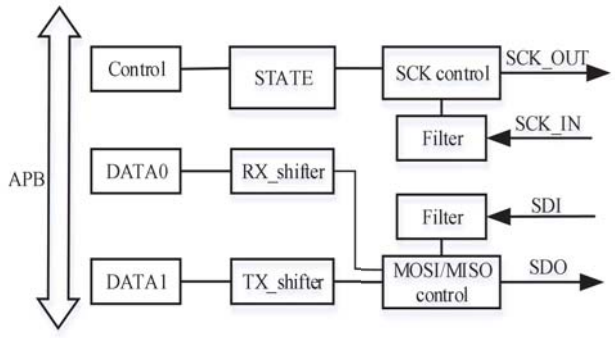

Figure 2. SPI architecture diagram

B. Block Diagram

According to the relevant requirements of the SPI protocol, this paper designs the internal detailed structure of the interface module as shown in Fig. 2. APB Interface is the connection bridge between the whole SPI interface control and the APB bus, and it is responsible for the data exchange with the APB bus[7]. Through the APB bus interface, the CPU can write the data that needs to be sent into the data register, and it can also read the received data from the data register. At the same time, the CPU can operate the corresponding control register and status register through APB bus[8-9].

As shown in Fig. 2, DATA1 is a register of writing data, used to store the data that needs to be sent. DATA0 is a register of reading data, used to store the data received by SPI. The state machine STATE is the core part of the whole SPI control, generating control logic to control the receiving and sending of data. The serial receiving or sending of data is realized by the receiving shift register RX_shifter or the sending shift register TX_shifter. Filter is the filter circuit module of the input signal. When data is input to the SPI port or SCK is input, there may be glitch noise. If it is not filtered, the received signal may appear metastable. If it is used directly, it may cause errors in the data transmission result.

When the SPI receives data, the data MISO is input in the master mode, and the data MOSI is input in the slave mode. After the filter circuit, the data is serially sent to the receiving shift register. Finally the data stored in the data register DATA0. When sending data, firstly write the data in the writing data register TABLE I into the sending shift register, and then send the data serially under the control of the state machine.

C. SPI Register

The SPI interface module control register definition is shown in TABLE I. Before the communication of the SPI module, the SPI transmission form can be controlled by configuring these control registers. In the SPI control register, Mode is the master-slave mode selection bit; Shift_ctrl is the high and low selection bit of SPI transmission; CPOL is the clock polarity selection bit; CPHA is clock phase selection bit; CS_ctrl is the high and low level valid bits of the CS chip selection signal; SPI_workon is the start signal bit of SPI module working; Trans_max is the transmission length selection bit, 5'b00000-5'b11111 respectively indicate the transmission length from 1bit to 32 bit; SCK_SEL is the clock selection bit, which can change the transmission rate of SCK. The highest transmission clock frequency is 5M. Before starting the transmission, APB will configure all the control bits in the control register, and then according to the register configuration, the SPI interface will perform data transmission.

D. Design of State Machine

The data receiving and sending of the SPI module is controlled by the state machine. The SPI module of this design can be configured as a master or a slave by the control register.

When the SPI is configured as a master, the master state machine is shown in Fig. 3. Firstly, when there is no data transmission, the SPI module is in the IDLE state. Before SPI starts to work, it is necessary to configure the master mode, transmission word length, and clock transmission polarity in IDLE state. Then pull down the CS_OUT signal (the default value of the CS signal is active low) and configure the workon (start transmission) signal. At this time, the state machine jumps from the IDLE state to the WORKING state. In this state, the SPI starts to generate the clock signal SCK_OUT, and Send or receive data from the data line. When all the data is transmitted, the SPI transmission done signal is generated. Then the state machine jumps to the FINISHI state. In the FINISH state, it returns to the IDLE state immediately after 1T. At this time, a complete SPI master transfer ends, and SCK returns to the default state.

TABLE I. THE SPI CONTROL STATUS REGISTER DESCRIPTION

Width	Name	Dscription
1bit	Mode	Mode Flag 1:master mode 0 : slave mode
1bit	Shift_ctrl	Shift control 1: MSB 0: LSB
1bit	CPOL	SCK Polarity 1: SCK idle level high 0: SCK idle level low
1bit	CPHA	Clock phase 1:The second sck is valid 0:The first sck is valid
1bit	CS_ctrl	0:Low level active 1:high level active
1bit	SPI_workon	SPI starts working
5bit	Trans_max	Transmission word length selection
2bit	SCK_SEL	SCK transmission clock selection

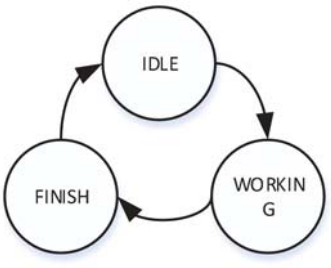

Figure 3. SPI master state machine

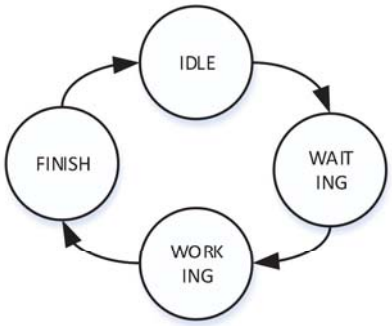

Figure 4. SPI slave state machine

When the SPI is configured as a slave, the slave state machine is shown in Fig. 4. When SPI is not working, the slave state machine is in the IDLE state, waiting for the start condition. Similar to the master operation, the working mode, transmission clock polarity, and transmission length must be configured firstly. When configured in slave mode and the WORKON signal is valid, the state machine enters the WAITING state. In this state, it waits for the master to initiate a transmission signal (CS pulls low) for data transmission. When CS pulls low is detected, the state machine enters the WORKING state. In this state, the clock SCK_OUT output by the master is received, and data reception or transmission starts. When it is detected that CS is pulled high, it indicates that the data transmission is completed and the SPI transmission completion signal is generated. At this time, the state machine enters the FINISH state. Then it returns to the IDLE state, which indicates the end of a complete SPI slave transfer.

III. Simulation and Verification

This design is implemented by the Verilog HDL. After the RTL code is completed, ModelSim is used to simulate the designed circuit at the RTL level. Through comprehensive simulation, it is verified that the circuit logic function is consistent with the expected function. However, the logic function verification is only to verify the logic implemented by the hardware description language, and cannot explain whether the actual circuit is correct. Therefore, after the RTL level verification is completed, in order to ensure that the function of the described circuit is completely correct, we adopt the FPGA verification method[10]. In FPGA verification, the designed SPI is imported into FPGA and communicated with the external SPI device to verify the correctness of the design.

A. RTL Simulation Verification

Fig. 5 shows the simulation waveform of the SPI master mode. The transmission length of the SPI transmission setting is 8 bits, the transmission polarity is 0, and the configured transmission data is 5A. As can be seen from the figure, the SPI master device can correctly generate SCK and chip select signal CS_OUT. It can send 5A correctly and generate SPI transmission completion signal after data transmission is completed. As shown in Fig. 6 and Fig. 7, the simulation waveforms of sending and receiving data in SPI slave mode. It can be seen from the figure that the SPI slave can send data 5A or receive data AD69 correctly. The state machine jumps

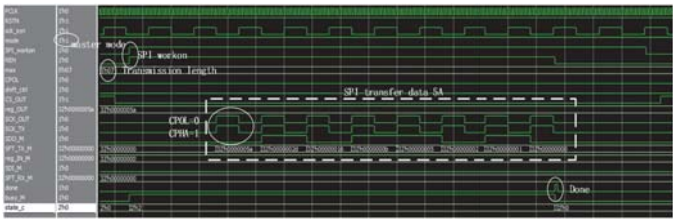

Figure 5. SPI master transmission simulation waveform

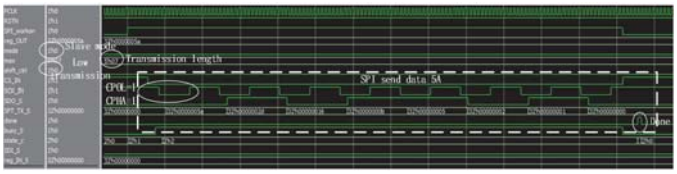

Figure 6. SPI slave sending data simulation waveform

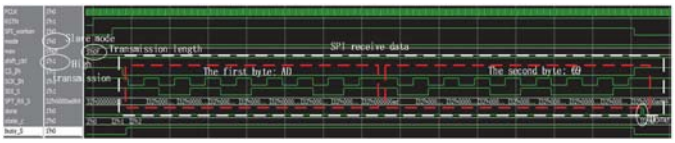

Figure 7. SPI slave receiving data simulation waveform

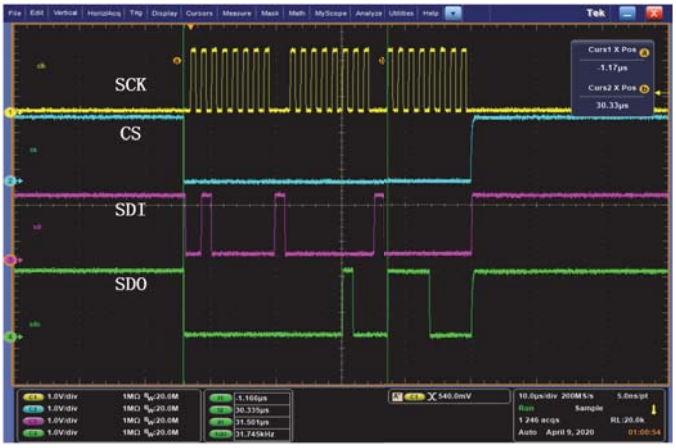

Figure 8. SPI FPGA verification diagram

correctly during transmission. There is no transmission error, and the completion signal is correctly generated after the transmission is completed. By comparing the above three simulation diagrams, the results show that the SPI interface designed in this paper is fully functional, with the advantages of configurable master/slave mode, setting the transmission direction, changing the transmission rate, changing the data transmission length and being suitable for 4 kinds of clock modes. The functions of the SPI interface based on the APB bus can be realized.

B. FPGA Verification

In the FPGA verification stage of the SPI interface, RTL is imported into Vivado for processing, and the generated BIT file is imported into the FPGA test board. Test the output pins of the FPGA. Configure SPI as master mode, set SCK clock polarity to 0, and perform joint debugging through external SPI slave devices. If the transmission length is 8 bits of data, the test result is captured by the oscilloscope as shown in the

Fig. 8. The result shows that the SPI output SCK is normal, and the data can be transmitted according to the configured working mode. The designed SPI module can communicate with an external SPI slave device. The designed circuit functions are correct.

IV. CONCLUSION

This paper designs and implements a configurable SPI interface based on the APB bus. Through configuration registers, this SPI interface supports master-slave mode selection, four kinds of clock transmission modes, two modes of most significant bit and least significant bit, variable transmission length from 1 bit to 32 bits, and optional transmission rate. This design uses the finite state machine design method and the Verilog HDL for logic implementation. Finally, through RTL simulation and FPGA verification, the designed SPI can communicate correctly with other SPI devices.

ACKNOWLEDGMENT

This work was supported by the project (No.5100-201941433A-0-0-00) of the State Grid Corporation of China in 2020 “Research on Key Technologies and Sample Development of Low-end MCU in the Field of Relay Protection”.

REFERENCES

[1] F. Leens, "An introduction to I2C and SPI protocols," in IEEE Instrumentation & Measurement Magazine, vol. 12, no. 1, pp. 8-13, February 2009.

[2] Z. Xin, H. Lu, L. Hu and J. Li, "Implementation of SPI and driver for CC2430 and C8051F120," 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang , pp. 2638-2641,2012.

[3] H. H. Lu, "Design and realization of SPI driver based on VxWorks," 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, pp. 200-205, 2020.

[4] A. N, G. Joseph, S. S. Oommen and R. Dhanabal, "Design and implementation of a high speed Serial Peripheral Interface," 2014 International Conference on Advances in Electrical Engineering (ICAEE), Vellore, pp. 1-3, 2014.

[5] M. Hafeez and A. Saparon, "IP core of Serial Peripheral Interface (SPI) with AMBA APB interface," 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Malaysia, pp. 55-59, 2019.

[6] A. K. Oudjida, M. L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui and Y. N. Alhoumays, "Design and test of general-purpose SPI master/slave IPs on OPB bus," 2010 7th International Multi- Conference on Systems, Signals and Devices, Amman, pp. 1-6, 2010.

[7] L. Bacciarelli, G. Lucia, S. Saponara, L. Fanucci and M. Forliti, "Design, testing and prototyping of a software programmable I2C/SPI IP on AMBA bus," 2006 Ph.D. Research in Microelectronics and Electronics, Otranto, pp. 373-376, 2006.

[8] B. Qu and D. W. Fan, "Design of remote data monitoring and recording system based on ARM," 2010 2nd International Conference on Industrial and Information Systems, Dalian, pp. 252-255, 2010.

[9] J. L. Zhang, C. Y. Wu, W. J. Zhang and J. W. Wang, "The design and realization of a comprehensive SPI interface controller," 2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot, pp. 4529-4532, 2011.

[10] S. Saha, M. A. Rahman and A. Thakur, "Design and implementation of SPI bus protocol with Built-in-self-test capability over FPGA," 2014 International Conference on Electrical Engineering and Information & Communication Technology, Dhaka, pp. 1-6, 2014.