Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 5, No. 1, 2023

Design and Implementation of IIC Interface IP Core

Jincheng Huang

Southwest Minzu University, Sichuan, China

Abstract: With the development of SOC (System on Chip) design technology, IPcore reuse technology and on-chip bus technology, which are the basis of SOC design technology, have attracted wide attention from the society. IP core reuse technology can save repetitive work and greatly speed up chip development. At present, China's large-scale SOC design uses more foreign IP cores, which are expensive. IIC(Inter-integrated Circuit) interface is an important peripheral interface in AMBA (Advanced Microcontroller Bus Architecture) bus. It is used for communication between CPU and peripheral devices, and can be used in embedded, radio frequency communication and other fields. Therefore, designing an IIC interface IP has great use value.

Keywords: IP core, IIC interface, SOC technology.

1. Introduction

The integrated circuit industry is the core of the information technology industry. It is a strategic and basic pilot industry that supports social and economic development and national strategic security. When the scale of the integrated circuit develops to the point where the entire digital computer system can be integrated into one chip, the system chip (SOC) is born. The system chip usually refers to the computer system implemented on a single chip [1]. IIC bus can control IO extenders, various sensors, EEPROM, AD/DAS and other devices through only two signal lines, which is also the most advantage of IIC bus compared with other protocols. Because the interface is directly on the component, the IIC bus occupies very little space, reducing the space of the circuit board and the number of chip pins, and supporting multiple master control. Any device capable of sending and receiving can become the main line. The IP core of IIC bus interface has been increasingly integrated into SOC. At present, in the large-scale SOC design in China, compared with foreign countries, there are very few available IP cores, so it is of practical significance to design a reusable IP core [2].

2. Technical Route

The technical block diagram is shown in Figure 1, which is mainly divided into internal register module, APB interface module, FIFO module and serial-parallel conversion module.

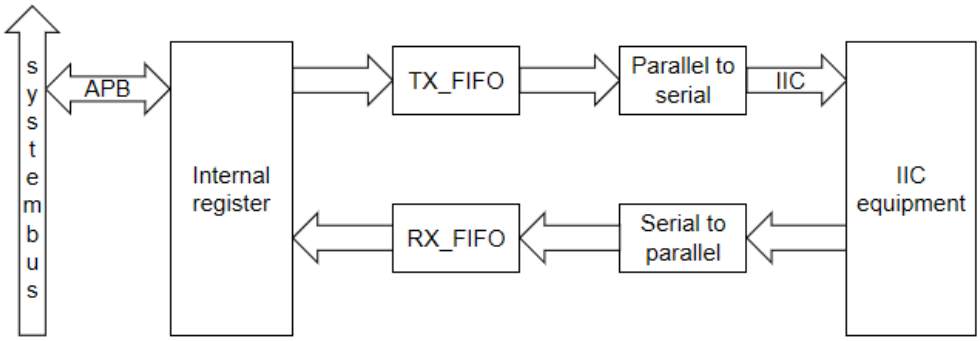

Figure 1. The technical block diagram

2.1. Internal register module

The functions of IIC interface such as data transmission and function selection are realized by configuring many registers. According to the functions required by the specification, relevant registers are designed, including the configuration register (IIC_CON), interrupt register (INT_FIAG), receive data register (RX_DATA) and send data register (TX_DATA). The specific functions are shown in Figure 2.

Figure 2. Register function

Name	Access	Dscription
IIC_CON	RW	IIC Configuration Register
INT_FLAG	RW	The Interrupt Status Register
RX_DATA	RW	Receive Data From IIC Flash
TX_DATA	RW	Transfer Data From CPU

2.2. APB interface module

In order to connect the CPU and various peripheral modules, APB bus is adopted here. The CPU and various peripheral devices are mounted on the bus, so that commands and data can shuttle freely between the CPU and peripheral devices. APB bus is a common bus standard defined by AMBA protocol. It is used to communicate with external devices and has the characteristics of low power consumption and simple interface [3]. Generally, some general serial ports are connected to the low speed APB bus. In this design, the specific function is to control the reading and writing of registers.

APB bus standard interface is shown in Figure 3.

Figure 3. APB bus standard interface

Name	attribute	width	description
PSEL	Input	1	Chip selection signal, high active
PWRITE	Input	1	Read and write enable signal, high is write effective, low is read effective
PWDATA	Input	32	Write data bus
PADDR	Input	5	Read and write address bus
PRDATA	Output	32	data bus
PENABLE	Input	1	Read and write transmission valid confirmation signal

2.3. FIFO module

FIFO is a first-in, first-out data buffer. The difference from ordinary memory RAM is that there is no external read/write address line, which is very simple to use; However, the disadvantage is that the data can only be written in sequence, and the data address of the sequential read data is automatically completed by adding 1 to the internal read/write pointer. It cannot be determined by the address line to read or write a specified address, as is the case with ordinary memory. The specific interface function of FIFO is shown in Figure 4.

Figure 4. The specific interface function of FIFO

Name	width	attribute	description
SYSCLK	1	INPUT	System clock
RST_B	1	INPUT	System reset
WR_EN	1	INPUT	Fifo write enable input
RD_EN	1	INPUT	Fifo read enable input
FIFO_IN	8	INPUT	Fifo write data input
FIFO_OUT	8	OUTPUT	Fifo read data output
EMPTY	1	OUTPUT	Fifo empty output
FULL	1	OUTPUT	Fifo empty output

When the data is transferred from the bus or peripheral device, the register starts to read and write. At this time, the data will not be transferred immediately, but two FIFOs are set in the middle to store the data. At this time, the FIFO is empty and full. If it is not empty, it means there is data, and the data can be transferred.

The main difficulty of FIFO design is to judge the empty and full flag of FIFO. Here we will briefly describe the algorithm of judging the empty and full state of FIFO. Assuming that the depth of FIFO is 2 ^ N bytes, construct a pointer with a width of N+1 read and write pointers. When the highest bit of the binary code of two pointers is inconsistent and the other N bits are equal, the FIFO is full. When the pointers are completely equal, the FIFO is empty. In this way, the empty and full status of FIFO can be determined with very simple logic. In this design, the depth of the two FIFOs is 4, so the width of the write pointer WP and the read pointer RP is 3. The two pointers follow the principle of first performing the read and write operation and then adding 1. In this design, WR__ PTR is the write pointer of the FIFO. The FIFO is written externally once, WR__ PTR plus 1, RD__ PTR is the reading pointer, external FIFO is read once, RD_ PTR plus 1. When WR_ PTR= RD_ _ PTR indicates that FIFO is empty at this time; When WR_ PTR [2]=(~RD_ PTR [2]) and WR_ PTR [1:0]= RD_ PTR [1:0] indicates that FIFO is full at this time.

2.4. IIC interface module

(1) Interface definition and overall design of IIC transmission module

In addition to the two signal lines (SCL and SDA) for IIC communication, the transmission module also includes some clock, reset, enable, parallel input and output and completion flag bits. The block diagram is shown in Figure 5.

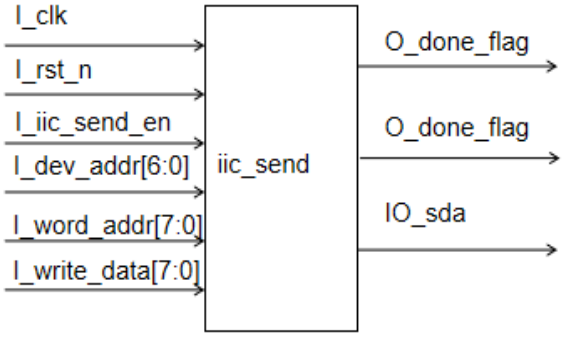

Figure 5. IIC sending module

I_ Clk is the system clock;
I_ rst_ N is system reset;

I_ iic_ send_ En sends enable signal when I_ iic_ send_ When en is 1, the IIC master (FPGA) can send data to the IIC slave;

I_ dev_ Addr [6:0] is the device address of the IIC slave;

I_ word_ Addr [7:0] is the word address, which is the internal storage address of the IIC device we want to operate;

I_ write_ Data [7:0] is the data to be written by the host (FPGA) to the IIC word address;

O_ done_ Flag is a byte completion flag bit sent by the host (FPGA), and a high pulse will be generated after the transmission is completed;

O_ Scl is the serial clock line of IIC bus;

IO_ Sda is the serial data line of IIC bus;

To realize iic_ The function of the send module divides the IIC data transmission process into the following 10 states;State 0: idle state, used to initialize the value of each register

State 1: Load the physical address of IIC device

State 2: Word address of IIC device loaded

Status 3: Load the data to be sent

State 4: send start signal

Status 5: Send a byte, starting from the high bit

Status 6: Receive the response bit of the response status

Status 7: Check the response bit

Status 8: Send stop signal

Status 9: IIC write operation ended

The above states are in no particular order. Some states need to be reused multiple times.

(2) Interface definition and overall design of IIC receiving module

In addition to the two signal lines (SCL and SDA) for IIC communication, the IIC receiving module also includes some clock, reset, enable, parallel input and output and completion flag bits. The block diagram is shown in Figure 6.

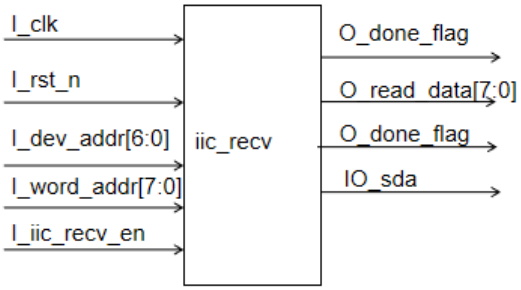

Figure 6. IIC receiving module

I_ Clk is the system clock;

I_ rst_ N is system reset;

I_ iic_ recv_ En receive enable signal when I_ iic_ recv_ When en is 1, the IIC host (FPGA) can receive data from the IIC slave;

I_ dev_ Addr [6:0] is the device address of the IIC slave;

I_ word_ Addr [7:0] is the word address, which is the internal storage address of the IIC device we want to read;

O_ read_ Data [7:0] is the data read by the host (FPGA) from the IIC device word address;

O_ done_ Flag is the completion flag bit of a byte received by the host (FPGA). After receiving, a high pulse will be generated;

O_ Scl is the serial clock line of IIC bus;

IO_ Sda is the serial data line of IIC bus;

According to IIC timing analysis, the process of receiving a byte of data has a second start signal and control byte (CONTROL BYTE) compared with sending a byte of data, and the lowest bit of the second control byte (CONTROL BYTE) should be 1, which means that the IIC master (FPGA) reads data from the IIC slave (24LC04). When the host (FPGA) wants to end the process of reading data, it will send a non-response bit 1 to the IIC device, Finally, the whole process of reading data is ended by sending a stop signal. Therefore, according to this process, the following states can be summarized:

State 0: idle state, used to initialize the value of each register

State 1: Load the physical address of IIC device

State 2: Word address of IIC device loaded

State 3: send the first start signal (the reading process requires sending two start signals)

Status 4: send a byte of data, starting from the high bit

Status 5: response bit of receiving response status

Status 6: Check the response bit

State 7: send the second start signal (the reading process requires sending two start signals

Status 8: The physical address of IIC device is loaded again, but the last bit of the physical address this time should be 1, indicating read operation

Status 9: Receive a byte of data, starting from the high bit

Status 10: The host sends a non-response signal 1 to the slave

Status 11: After confirming that the slave receives the non-response signal 1, the SDA initialization value is 0 and is ready to generate a stop signal

Status 12: Send stop signal

Status 13: End of read operation

The above states are in no particular order. Some states need to be reused multiple times.

2.5 Serial-parallel conversion

Series-parallel conversion is often used in the field of high data transmission. Serial transmission is to convert the parallel data to be transmitted inside the computer into serial data, transmit it through a communication line, and convert the received serial data into parallel data.

Before the computer sends data serially, the parallel data inside the computer is sent to the shift register and moved out bit by bit to convert the parallel data into serial data, as shown in Figure 7 below.

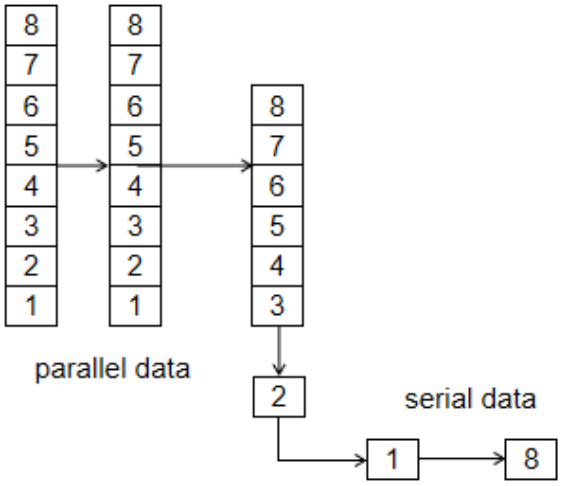

Figure 7. Serial-to-parallel conversion during transmission

When receiving data, the serial data from the

communication line is sent to the shift register, which is stored in 8 bits and then sent to the computer in parallel, as shown in the figure 8 below.

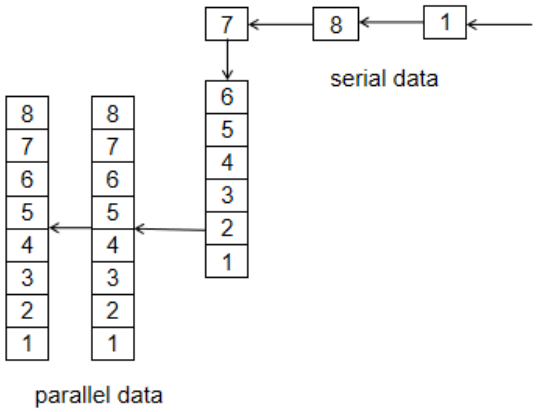

Figure 8. Serial-to-parallel conversion upon reception

3. Conclusion

After completing the above modules, write a top-level module to integrate them together, and complete the design and implementation of the IP core for the IIC interface. Compared with other IP cores, the IP core designed in this paper takes less logical resources and has complete functions. It can be directly attached to the SOC for use, which is very convenient.

References

[1] He Shan, Design and Verification of IIC Bus Interface IP Core, Hefei Polytechnic Universit,2007.

[2] Wang Dawei, Design and Verification of SPI Interface IP Core Based on UVM, Northern Polytechnic University,2022.

[3] Qin Yu, Design and verification of SPI interface IP core based on APB bus,Guizhou University, 2016.