Both Common Techniques for The Design of I2C Bus Contdroller and Compare between Two Techniques

Xu ning Chen Wei Li yu hui
CTI lab, Kunming University of Science an Technology,
Kunming 650051
ffaningff@163.com

Abstract

I2C-bus is applied widely in many practical systems, so we have designed I2C-bus controller by MCU and FPGA/CPLD .Two techniques are in common use. And I2C-bus controller designed by two techniques is steady and reliable .The designer will select one between both techniques, according different systems.

Keywords: I2C-bus, MCU,FPGA, FSM,VHDL

1. Introduction

In consumer electronics, telecommunications and industrial electronics, there are often many microcontroller, memorys, data converters, data tuners, DSP blocks and so on in many systems.

As to designs above, Philips developed a simple bi-directional 2-wire bus for inter-IC control, and to modularize and standardize electrocircuit systems. This bus is called I2C-bus. Via the I2C-bus, the electrocircuit systems become compacter and smarter to the benefit of both systems designers and equipment manufacturers. This design concept solves many interfacing problems.

Each device connected to the bus is software addressable by a unique address and simple master/slave relationships exist at all times; masters can operate as master-transmitters or as master-receivers. In I2C-bus, for transferring data, at first send START condition, then send valid data. But note that after transferring a byte there must be a acknowledgement bit (ACK). After transferring the last byte, there should be a STOP condition going with unacknowledgement(NACK).

As showing Figure 1:

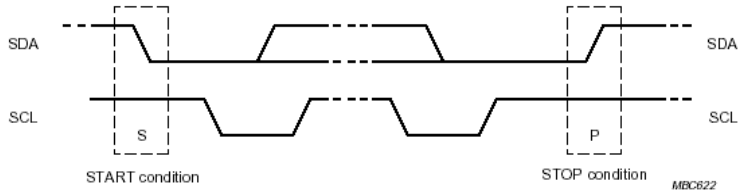

Fig.1

For designing I2C-bus we will supply two methods here.

2. FPGA/CPLD project for I2C-bus design

AS to this project, we supply an explicit example that via I2C-bus we realized the configuration of SAA7128 (a video encoder of Philips). Our main mission is the design of FSM(finite state-machine).The key of the design is that you can control the timing scheduling and attemper every function block exactly. So it is an advisable method using FSM for this design. And we have adopted top-to-down design which is a classical technique for IC design.

2.1 The top block

According to datasheet of SAA7128, we have the format of I2C-bus below:

S	SLAVE ADDRESS	ACK	SUBADDRESS	ACK	DATA 0	ACK	-------	DATA n	ACK	P

You can see that the sending signal ordinarily is: START condition, Slave address of I2C-bus, acknowledgement, the Subaddress , acknowledgement, the first effective data……,and STOP condition. So we will design a FSM for sending data. In view of SAA7128 embraceing too many registers (count up to 128), we can't use a state for every byte to transfer every data, so we will optimize the VHDL code via "state reuse". And we should define all sending data as an array. The state diagram is figure 2.

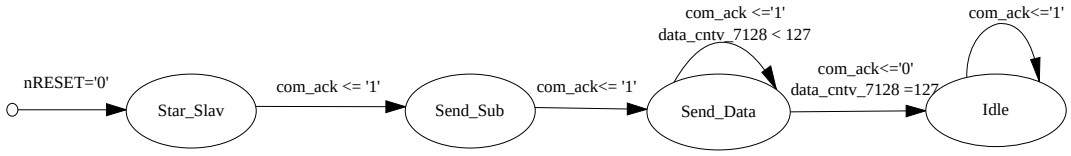

Fig.2

As is showing in fig.2,it is clear that there only are 4 states for sending all the data. They are slave address, subaddress, effective data and idle state. And you can find the "state reusing" for sending data.

2.2 Command shift FSM

According to fig.3, you can see 6 states for command shift FSM. They are St_idle, St_start, St_write, St_read, St_ack and St_stop. At the beginning of data transfer, the system is reseted, and then becomes at idle state. After the system has received GO command and START command, it will come into Start state, then by judging the latter command .If that is write command, it will come into WRITE state, vice versa. You should note a "dcnt" signal. It is used to counting for bit of a byte. When finished a data transfer, the system will be at ACK state which indicates a transfer work has finished. In succession, if the system receives Stop command, it will be at St_stop state, otherwise come into St_Idle state and be ready to read/write the latter byte or return start state once again. Now you should be conscious of the main function of the FSM that is startuping down I2C-bus timing FSM and shifting the data in array into SDA line serially .In our design, we don't startup St_read state, as our aim is to write the register of SAA7128, but here we reserve St_read state for the integrality of I2C-bus controller.

2.3 Down timing FSM

For the design of down timing FSM, you can use 5 child state like A,B,C,D and IDLE to respectively denote Start state, read/write state and Stop state. According to I2C-bus specification, LOW period of the SCL clock should above 4.7 us, so we must do a divider of main clock. This must be designed according to your clock. The FSM is shown in fig.4.

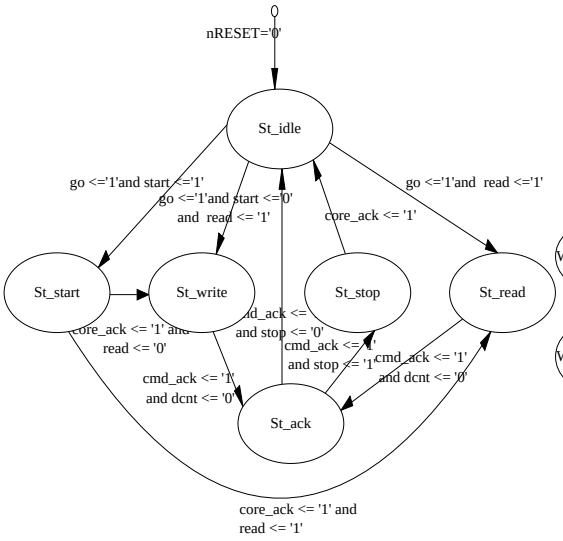

Fig.3

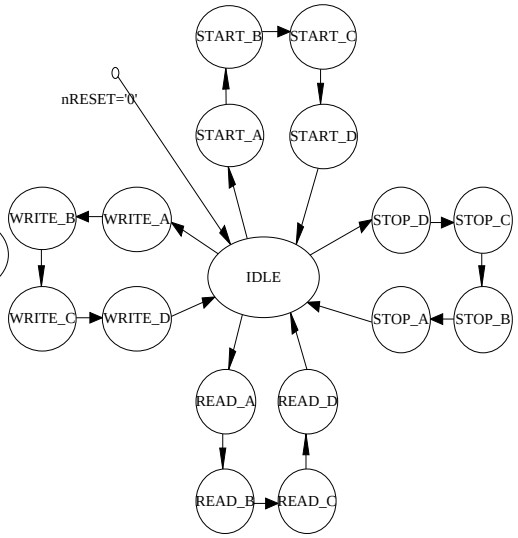

Fig.4

We have used Spartan2 of Xilinx to realize, VHDL to describe, ModelSimSE to simulate and Synplify Pro7.6 to synthetize.Fig.5 shows the part result of ModelSimSE:

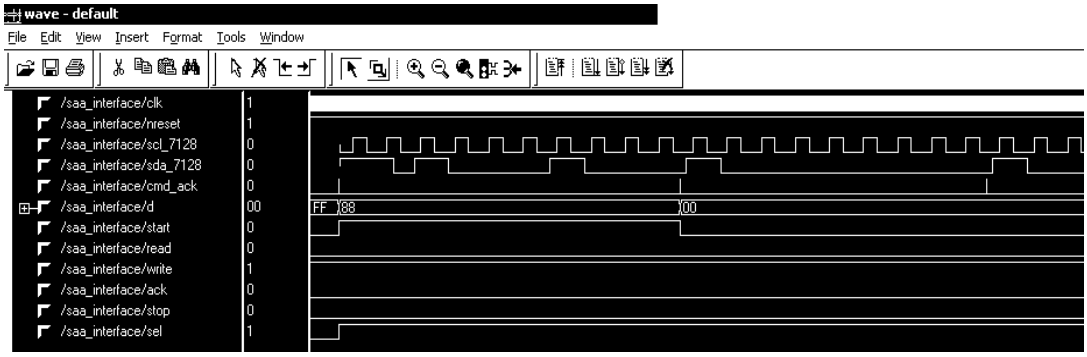

Fig.5

3 MCU project for I2C-bus design

In this project, the MCU is W77E58 of Winbond.

3.1 Start Condition Block

Look into fig.1, we can find that during the High of SCL clock, SDA becomes Low period from High period. According to the characteristic, we make the period hold for a while by delaying in software. This must be in the light of machine cycle of MCU and I2C-bus specification to establish NOP command. The Start condition block is as bellows:

```
START：  SETB    SDA
         NOP
         NOP
         NOP
         ……
         CLR   SDA
         NOP
         NOP
         NOP
```

……
CLR
SCL
RET

3.2 Transfer Data Block

As to the design of this block, here we see sending data block as a example. And we should note the acknowledgement bit after every byte. Part of the code is as bellows:

```
SE_DATA：
    CLR      A
    MOVC     A,@A+DPTR
    MOV      DATA1,A
    CALL     OUTBYT
    CALL     ACK
    INC      DPTR
    DJNZ     R0,SE_DATA
```

By logic analyzer, we can see the wave is as fig.6:

Notes in fig.6: J3.1:0 is SCL,J3.0:0 is SDA .The first sending signal is start condition, then is the byte 01001000(48H).

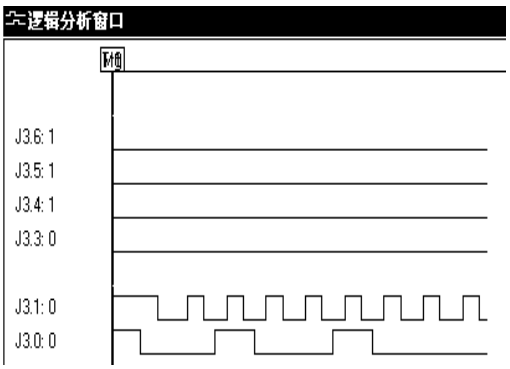

Fig.6

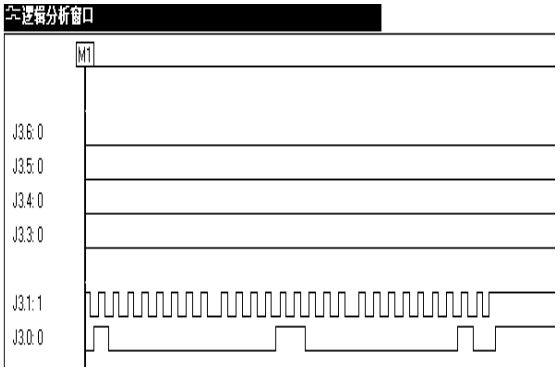

Fig.7

3.3 Other Block

As to the design of ACK block, NACK block and STOP block, the method is similar to of Start Condition Block .The whole simulate wave is shown in fig.7.Notes in fig.7: The last sending data is 00000001(0H)and the transfer task is finished by Stop Condition.

4 The compare between both methods for designing I2C-bus controller

According to the process of the designs, we can get the difference between them as bellows:

At first, when we use FPGA to do our task, we use FSM to realize the transfer of data and simulate every state of I2C-bus.During transfer process, we must set many signals, and exactly control the flow of them. Secondly, when we layout the whole design, there may be some delay which maybe result in the failure of data transfer, so we must modify and improve the code continually, and do function simulate ,synthesis, layout and timing simulate to make sure the veracity of data transfer. The task is comparatively oppressive.

But when we use MCU to gain our ends, the process is very easy. The problems witnessed are much less. The reason is that MCU simulates every state of I2C-bus and controls the data transfer via special instruction NOP, no emerging timing disorder.

5.Conclusion

At last, we have designed I2C-bus controller by both techniques, and gotten sound and stable outcome. And the fruit realized by FPGA has been embedded in our video process project. The designer can select appropriate method to meet own request. For instance, in the system where MCU is main controller, you can use FPGA/CPLD to do I2C-bus controller for save the resource of MCU. But in the system where FPGA is the key, if FPGA possesses of enough resource, you can choose FPGA for controlling I2C-bus, and the whole system will become much smarter and compacter.

Reference

[1] Zhou Li Gong etc, “MCU and CPLD application. Beijing”, Publishing Company of Beijing University of Aeronautics & Astronautics ,2003.8

[2] Hu Zhen Hua, “VHDL&FPGA design.Beijing”, Publishing Company of China Railroad ,2002.12

[3] I2cspecification Version2.1 . Philips Semiconductors,2000.1