Porting from Wishbone Bus to Avalon Bus in SoC Design

Xu Xing Chen Zezong Jiang Jing Ke Hengyu
(School of Electronic Information, Wuhan University, Wuhan 430079, China)

Abstract: Motivated by the increasing IP (Intellectual Property) based SoC (System-on-chips), designers have begun using an IP based SoC design methodology that permits reuse of key SoC functional components. The Wishbone bus is a common interface between IP cores, and the Avalon interface is designed to accommodate peripheral development for the SoC environment. It is necessary to port the Wishbone bus to Avalon bus when we use an IP core with a Wishbone interface in an Avalon bus system. We port the Wishbone interface I2C controller IP to Avalon bus and design a master/slave simulation model to test the Avalon bus compatible I2C controller IP core. The experimental results confirm that the Avalon bus compatible I2C controller IP works well in the Avalon based SoC.

Keywords: SoC; IP; Avalon Bus; Wishbone Bus; Switch-fabric.

1 Introduction

As billion transistors SoC become commonplace and design complexity continues to increase, designers are faced with the daunting task of meeting escalating design requirements in shrinking time-to-market windows and have begun using an IP based SoC design methodology that permits reuse of key SoC functional components[1]. OENCORES.ORG [2] provides many Wishbone compatible IP cores. We have to port the I2C IP core from Wishbone bus to Avalon bus in our SoC Design.

The Wishbone SoC interconnection architecture for portable IP cores is a flexible design methodology for use with semiconductor IP cores, with a purpose to foster design reuse by alleviating SoC integration problems. This is accomplished by creating a common interface between IP cores. It improves the portability and reliability of the system, and results in faster time-to-market for the end user[2].

The Avalon interface is designed to accommodate peripheral development for the SoC environment. The Avalon defines transfers between a peripheral and a switch-fabric interconnect structure. The Avalon's interconnect strategy allows system designers to connect any master-type peripheral to any slave-type peripheral without a prior knowledge of either the master or slave interface. The configurable interconnect strategy allows a peripheral designer to limit the signal types needed to support the specific type(s) of transfers desired [3].

The paper focuses on porting Wishbone bus compatible I2C controller IP (shorted as Wishbone I2C IP) to Avalon bus. We analyze the differences and similarities between the two buses and provide a way to port Wishbone I2C IP to Avalon compatible I2C IP (shorted as Avalon I2C IP). In order to evaluate the Avalon I2C IP, We design a master/slave model and test it in Modelsim. The Avalon I2C IP is mapped to Cyclone II FPGA as peripherals of Avalon bus in our SoC.

2 Wishbone and Avalon Bus

2.1 Wishbone Bus Features

The Wishbone interconnection makes SoC and design reuse easily by creating a standard data exchange protocol. Features of this technology include[2]:

(1) Full set of popular data transfer bus protocols including: READ/WRITE cycle, BLOCK transfer cycle, RMW (read-modify-write) cycle.

1-4244-1135-1/07/$25.00 ©2007 IEEE.

(2) Variable core interconnection methods support point-to-point, shared bus, crossbar switch, and switch-fabric interconnections.

(3) Master/slave architecture for very flexible system designs.

(4) Arbitration methodology is defined by the end user (priority arbiter, round-robin arbiter, etc.).

(5) 64-bits maximum data bus width.

2.2 Avalon Bus Features

Some of the prominent features of the Avalon interface are[3]:

(1) Separate Address, Data and Control Lines. Do not need to decode data and address cycles.

(2) Synchronous Operation. It simplifies the timing behavior of the Avalon interface and facilitates integration with high-speed peripherals.

(3) Dynamic Bus Sizing. Avalon bus supports data paths up to 128 bits. Avalon peripherals with different data width can interface easily with no special design considerations.

2.3 Comparison of Both Bus

We make the comparison between the two buses in respect of interconnection ways, signals, and operations.

(1) IP Cores Interconnection Ways: Wishbone bus has many interconnection ways, such as switch-fabric interconnection. And Avalon supports switch-fabric interconnection too. Peripherals and processor are interconnected via a switch-fabric bus. Fig 1 summarizes the SoC based on Wishbone or Avalon bus.

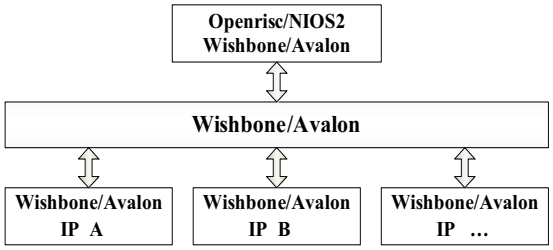

Fig 1 Switch-fabric interconnect structure

(2) Signals of Slave Device IP core: For both Wishbone bus and Avalon bus, the master signals are different from slave signals. Slave devices are more often used in practice.

These slave signals are divided into reset, clock, data and address, control (byte select, chip select, read, write. etc.), interrupt etc. as shown in Table 1.

Table 1 Signal of Wishbone Bus and Avalon Bus

	Wishbone Bus	Avalon Bus
Reset	rst: Synchronous reset	reset: Asynchronous reset
	arst: Asynchronous reset	
Clock	clk: Coordinates activities	clk: Synchronization clock
Interrupt	inta: Interrupt request	irq: Interrupt request
Data	dat_i: Data bus, input	readdata: Read data
	dat_o: Data bus, output	writedata: Write data
Address	adr: Address bus	address: Address
Control	sel: Byte(s) select input	chipselect: Chip select
	stb: Strobe input	read: Read-request
	cyc: Bus valid	write: Write-request
	we: The write enable input	byteenable: Enable byte lane
Handshake	ack: Termination bus cycle	Waitrequest: Stall the bus

3 Porting From Wishbone Bus to Avalon Bus

Every slave device could be divided into two parts: bus interface (master or slave) and function unit. The Wishbone bus interface of I2C controller is defined in module i2c_master_top (Wishbone signals are prefixed with wb_) [4][5].

```
module i2c_master_top
(
//Wishbone bus interface
wb_clk_i, wb_rst_i, arst_i, wb_adr_i, wb_dat_i,
```

wb_dat_o,wb_we_i, wb_stb_i, wb_cyc_i, wb_ack_o, wb_inta_o,

//device interface

scl_pad_i, scl_pad_o, scl_padoen_o, sda_pad_i, sda_pad_o, sda_padoen_o);

These Wishbone bus signals are matched in the following way:

(1) Reset signals: wb_rst_i, arst_i. I2C controller is an asynchronous reset device, we hold wb_rst_i at 0. Avalon has an asynchronous reset signal. reset_n, arst_i and reset_n are of the same function.

(2) Clock signals: wb_clk_i signal is the same with the synchronization clock clk of Avalon.

(3) Data and Address: wb_dat_i, wb_dat_o, wb_adr_i. The widths of Wishbone's data and address signals do not exceed the maximal size of Avalon. They can be connected to readdata, writedata, and address of Avalon bus.

(4) Control signals: wb_stb_i is device strobe signal of Wishbone, it is the same with chipselect of Avalon. wb_cyc_i indicates that the Wishbone cycle is enable, so it may be connected to chipselect. wb_we_i means Wishbone write enable and it may be connected to write signal of Avalon.

(5) Interrupt request signals: wb_inta_o is the same with Avalon's irq signal.

(6) Handshake signals: wb_ack_o. Connecting the wb_ack_o signal from the Wishbone bus to the Avalon bus is a bit more complex. When the slave device is ready to be read or written by the master device, the wb_ack_o signal will be held valid. We can connect it to the waitrequest_n signal of Avalon.

The top module of the I2C controller based on Avalon bus is as follows:

module i2c_master_top

(

clk, arst_i, address, writedata, readdata,

write, chipselect, chipselect, waitrequest_n, irq,

scl_pad_i, scl_pad_o, scl_padoen_o, sda_pad_i, sda_pad_o, sda_padoen_o);

4 Simulation

In the simulation model such as Fig 2, Avalon Master writes data (I2C data or I2C command) to the Avalon Slave device (I2C controller). After Avalon master sets the I2C controller at the working status, the I2C controller can read from or write to the I2C bus. Then we can monitor the data of the Avalon bus and I2C bus.

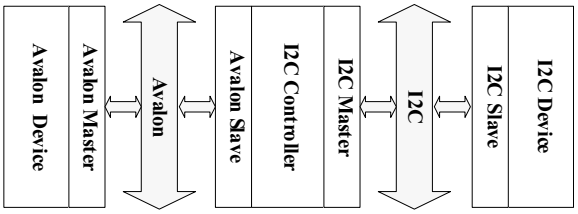

Fig 2 Master/Slave Simulation Model

Data sequence of the Avalon Master writes to the Avalon bus (hexadecimal): fa c8 00 80 20 90 xx… 01 10 xx… a5 10 5a 50….

fa c8 00 is written to frequency register.

80 is written to control register which enables I2C controller IP core.

When write data to I2C device for the first time, a start-write (90) command to the CR (command register) of I2C controller is to be sent. After that, write (10) command has to be sent to the CR before writing data to the TXR(Transfer) register[6].

The data sequence at Avalon bus is: Byte_1 90 byte_2 10 byte_3 10 …byte_n 10…byte_N 50. byte_n (n=1…N) is written to TXR register. 90 10 10 … 10 50 is written to CR register.

Data sequence written to the I2C bus is: 20 01 a5 5a….

Fig 3 illustrates the Avalon data bus flow of a simulation run. As shown in Fig 3, writedata sequence is: xx fa xx c8 xx 00 xx 80 xx 20 xx 90 xx…. xx stands for high resistance state. During reset and bus delay cycle, the data and address bus are in high resistance state. So the data appear at the Avalon bus are: fa c8 00 80 20 90…, the same with the data from the Avalon master device.

As illustrated in Figure 4, I2C data sequence is: s0010 0000 ar 0000 0001 ar 1010 0101 ar 0101 1010 ar…. s stands for start (sda change from 1 to 0). Each time we transfer a byte, we have to send an acknowledge signal (sda=0), shorted as ar indicates re-start (scl=0, sda=1). The valid bits of the I2C data

sequence are (hexadecimal): 20 01 a5 5a[7].

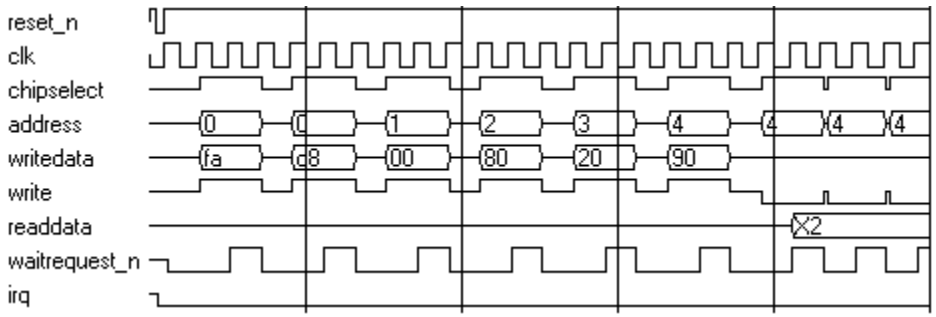

Fig 3 Avalon Slave Signals

Fig 4 I2C Signals

5 Conclusion

In addition to their inherent complexity, SoC designs also have to support the added burden of being flexible and quickly adaptable to meet changing needs of increasingly fickle customers. To meet these challenges, IP cores are designed modular and easy to be ported. In this paper, we introduce a way to port the I2C controller IP from Wishbone bus to Avalon bus in the SoC design. The Avalon I2C IP is mapped to Cyclone II FPGA as peripherals of Avalon bus. It has been proved by our experiments on NIOS2 development board that the Avalon I2C IP is synthesizable and has worked well.

Acknowledgment

This research was supported by NSFC (National Natural Science Foundation of China) award 60472012 and the 863 High Technology Project of China (2006AA09A303).

Reference

[1] S. Pasricha, N. Dutt, M. B. Romdhane,. Using TLM for Exploring Bus-based SoC Communication Architectures, ASAP, 16th IEEE International Conference, Page(s):79- 85, July 2005

[2] OPENCORES.ORG. WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.3, 2002.9

[3] Altera Corp. Avalon Interface Specification, 2005

[4] IEEE Computer Society. IEEE Standard Verilog® Hardware Description Language, IEEE Std 1364-2001, The Institute of Electrical and Electronics Engineers, Inc,28 September 2001

[5] M. D. Ciletti. Advanced Digital Design with the Verilog HDL, Publishing House of Electronics Industry, 2004.4

[6] R. Herveille. I2C-Master Core Specification, Rev. 0.9, 2003.7

[7] Philips Semiconductors, THE I2C-BUS SPECIFICATION, VERSION 2.1, 2000.1

Author Biographies

XU Xing is currently a M.S. student of Wuhan University in China. He is working on the research and development of System-on-chips, Embedded Operating System and Software.

CHEN Zezong was born in 1966. He received his Ph.D. degree in wireless physics from Wuhan University in 2005. He is a professor at the school of electronic information, Wuhan University. His research interests have included HF ground wave radar for ocean state remote sensing, software defined radio theory and Automatic Testing System.