Non-Transparent PCI-to-PCI Bridge Based on Verilog and FPGA

Zang Chunhua Shen Changli
College of Information Science and Technology
Nanjing University of Aeronautics and Astronautics
Nanjing, Jiangsu, China
E-mail: zang_ch@yahoo.com.cn

***Abstract*—PCI-to-PCI bridges are often used in CompactPCI systems to expand the PCI bus or to implement embedded and intelligent boards. The functions of the PCI-to-PCI bridge are briefly introduced. The implementation of a non-transparent PCI-to-PCI bridge on FPGA and the design of main modules are discussed. The result of timing simulation for the bridge is also provided.**

I. INTRODUCTION [3][4]

In the CompactPCI system，the PCI-to-PCI bridge is used to expand the PCI bus or implement an intelligent board. The bridge can be classified into transparent one and non-transparent one according to its operation mode.

For the transparent bridge, the devices on one side of the bridge are transparent to the devices on the other side of the bridge. The clocks on both sides must be synchronous. When a PCI transaction crosses the transparent bridge, the direct-through mode is used in the transfer of the transaction address that is not translated.

But for the non-transparent bridge, the devices on one side of the bridge are invisible to the devices on the other side of the bridge. The clocks on both sides can be independent. When a PCI transaction crosses the non-transparent bridge, the transaction address must be translated. This makes the two sides of the bridge independent processor domains. The processor on either side can independently configure and control its subsystem. In reality, the transparent bridge is used in the PCI bus expansion, while the non-transparent bridge is used in the intelligent PCI board.

One of the advantages for the implementation of the PCI-to-PCI bridge on FPGA is that it can be integrated into SOC or SOPC systems as an IP core.

The main difficulties of designing the bridge are system complication, high operation speed and electrical characteristics of the PCI bus, which can be solved by hierarchical design, optimization of system structure and logic design, and use of proper FPGA device respectively.

II. SYSTEM DESIGN [1][2][5]

The PCI-to-PCI bridge designed in this paper is used in the intelligent PCI board. Thus it is a non-transparent bridge. According to the function of the non-transparent bridge, the system structure is designed as shown in Fig.1.

The PCI I/O Module is for data acceptance and transmission and parity check specified by the PCI specification.

The Configuration & Status Space Module controlling the operation of the whole bridge consists of all the configuration registers and status registers in accordance with the PCI specification.

The Master_Target Unit and Target_Master Unit are key components of the system. The former realizes the transfer of PCI transactions from the secondary PCI bus to the primary PCI bus, while the latter does the same work in reverse direction. The design of the two units is the same, because the transaction transfer operations in two directions are identical. As space is limited, only the Master_Target Unit will be discussed in the ensuing paragraphs.

III. MASTER_TARGET UNIT DESIGN [1][2][5]

The main function of the Master_Target Unit is to let the PCI device on the secondary PCI bus take the initiative to

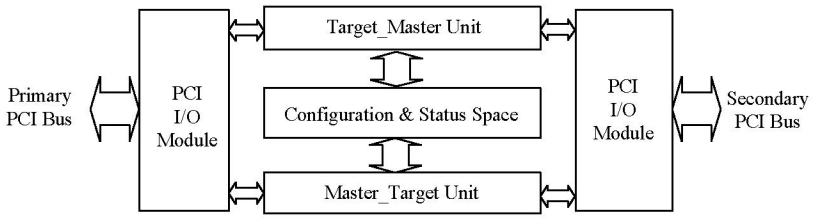

Figure 1. Structure of non-transparent PCI-to-PCI bridge

0-7803-9584-0/06/$20.00©2006 IEEE.

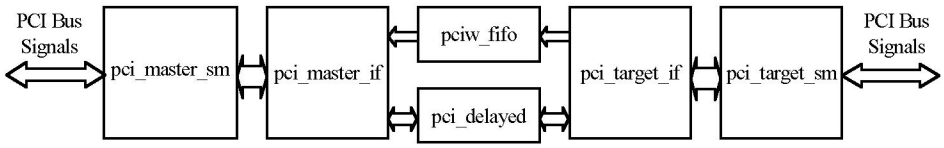

Figure 2. Block diagram of Master_Target Unit

visit the PCI device on the primary PCI bus. That is, the PCI device on the secondary PCI bus can originate a transaction to the PCI device on the primary PCI bus. For implementation of the function, The Master_Target Unit is designed as shown in Fig.2.

The pci_master_sm module is mainly to realize the state machine of the PCI master.

The pci_target_sm module is the state machine of the PCI target.

The pciw_fifo module is memory unit to transfer data between two PCI buses.

The pci_delayed module processes the delayed read transaction.

The pci_master_if module is the interface between the pci_master_sm module and the pci_delayed module as well as the pciw_fifo module， transferring data and transaction requests between the pci_master_sm module and the other two modules.

In similarity, the pci_target_if module is the interface between the pci_target_sm module and the pci_delayed module as well as the pciw_fifo module， transferring data and transaction requests between the pci_target_sm module and the other two modules.

The PCI-to-PCI bridge processes the PCI read command as the delayed read transaction.

A. Pci_target_sm Module

The pci_target_sm module realizes the PCI target state machine to accept PCI transactions. The target state machine is shown in Fig.3.

1) When the PCI-to-PCI bridge is reset, the target state machine is at the IDLE state.

2) When the target state machine is at the IDLE state and the FRAME# on PCI bus is active, it enters the ADDRESS state.

3) When the target state machine is at the ADDRESS state, it compares the accepted PCI transaction address with the base address. If the two addresses don't match, the target state machine returns to the IDLE state. Otherwise, some operations will be taken according to the type of the accepted transaction.

a) If the type is the PCI read transaction, the transaction address, bus command and byte enable signal are stored in the pci_delayed module through the pci_target_if module, the target retry is used to stop the transaction, and the target state machine returns to the IDLE state. At the meantime, the address on the other side of the bridge is translated and latched. When the read transaction on the PCI bus is retried, the target state machine compares the address, bus command and byte enable signal of the read transaction with the old latched values. If the two group values are identical and the data on the other side of the bridge are ready, the target state machine enters the TRANSFER state. Otherwise, the target retry is used to stop the transaction and the target state machine returns to the IDLE state.

b) If the type is the PCI write transaction, the target state machine goes to the TRANSFER state.

4) When the target state machine is at the TRANSFER state, it asserts DEVSEL# and TRDY# to transfer data. Meanwhile, if no transaction suspension and termination are detected, the target state machine will return to the IDLE state after the last data are transferred. If the transaction suspension is detected, the target state machine goes to the BACKOFF state. If the transaction termination is detected, the target state machine returns to the IDLE state.

5) When the target state machine is at the BACKOFF state and the PCI transaction suspension is cancelled within the PCI-specific time, it goes to the TRANSFER state again. Otherwise, it cancels the transaction and returns to the IDLE state.

When the target state machine conducts a PCI write transaction, the accepted transaction address is translated in light of the requirement of the PCI bus on the other side of the bridge, and the translated address and the accepted data are stored in the pciw_fifo module through the pci_target_if module.

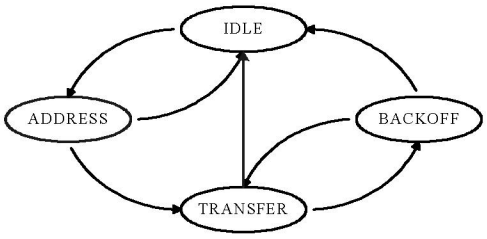

Figure 3. State transition diagram of state machine

B. Pci_master_sm Module

The pci_master_sm module realizes the PCI master state machine to originate (forward) PCI transactions. The state diagram of the PCI master state machine is the same as that of the PCI target state machine, except that the state transition conditions and operations are different.

1) When the PCI-to-PCI bridge is reset, the PCI master state machine is at the IDLE state.

2) When the master state machine is at the IDLE state and there is a delayed read transaction latched in the pci_delayed module or a write transaction stored in the pciw_fifo module, it detects whether the PCI bus is idle (whether both FRAME# and IRDY# are inactive). If the bus is idle, the master state machine asserts REQ# and detects whether GNT# is active. If GNT# is active, the master state machine enters the ADDRESS state.

3) When the master state machine is at the ADDRESS state, some operations will be taken according to the type of the originated (forwarded) transaction.

a) If the type is the PCI read transaction, the transaction message latched in the pci_delayed module is used to send transaction address, bus command and byte enable signal to the PCI bus.

b) If the type is the PCI write transaction, the transaction address, bus command and byte enable signal are fetched from the pciw_fifo module to originate the write transaction on the PCI bus.

When the PCI transaction target uses target retry to terminate the transaction, the master state machine waits several cycles to retry the transaction. If the target asserts DEVSEL# and IDRY# simultaneously to response the transaction within the PCI-specific time, the master state machine goes to the TRANSFER state. Otherwise, the transaction is terminated and discarded.

4) When the master state machine is at the TRANSFER state, it works as follows.

a) If the PCI read transaction is carried on, the data read from the PCI transaction target are latched in the pci_delayed module. Then the master state machine returns to the IDLE state.

b) If the PCI write transaction is carried on, the data stored in the pciw_fifo module are fetched out and sent to the PCI transaction target. If no transaction suspension and termination are detected, the master state machine will return to the IDLE state after transferring the last data. If the transaction suspension is detected, the master state machine goes to the BACKOFF state. If the transaction termination is detected, the master state machine returns to the IDLE state.

5) When the master state machine is at the BACKOFF state and the transaction suspension is cancelled within the PCI-specific time, it goes to the TRANSFER state again. Otherwise, it terminates the transaction and returns to the IDLE state.

IV. Implementation and Verification

A. Implementation

Verilog HDL and ALTERA Cyclone FPGA, EP1C12F324C6, are used to design the non-transparent PCI-to-PCI bridge. Quartus II and ModelSim SE are used as developing environment.

EP1C12F324C6 supports various I/O standards, including 66- and 33-MHz, 64- and 32-bit PCI standard. It contains 12,060 LEs, 239616 RAM bits, 8 distinct dedicated clocking resources and 249 maximum user I/O pins, meeting the requirement of designing the non-transparent PCI-to-PCI bridge. [6]

B. Testbench

After completing the logic design of the bridge, a testbench in line with the PCI specification is established, as shown in Fig.4.

During the simulation, the Master Device connecting with the primary interface of the bridge, originates PCI transactions, while the Target Device as the transaction receiver connecting with the secondary interface of the bridge, responds to the transactions. Clock, reset and arbitration module provides the clock and reset signals and the arbitrator of the two PCI buses. Because of the symmetry of the bridge, this testbench can verify both interfaces of the bridge.

C. Timing Simulation

As space is limited, only parts of simulation waveforms are provided and the idle cycles in the waveforms are compressed. The bridge is simulated at 32-bit data path and 33 MHz.

The write transaction is shown in Fig.5. All addresses and data are represented in radix 16. The first seven signals belong to the primary interface of the bridge, while the others belong to the secondary interface. The bridge responds the write transaction with address, 20000000, originated by the master device on the primary PCI bus.

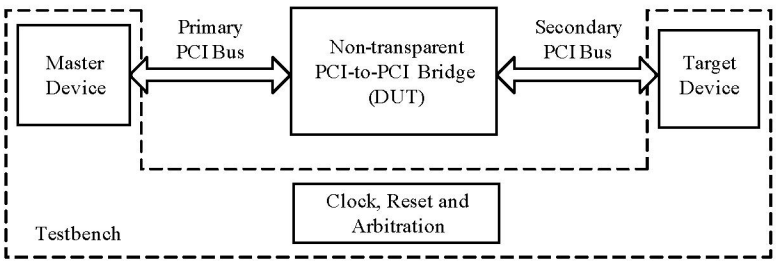

Figure 4. Testbench for non-transparent PCI-to-PCI bridge

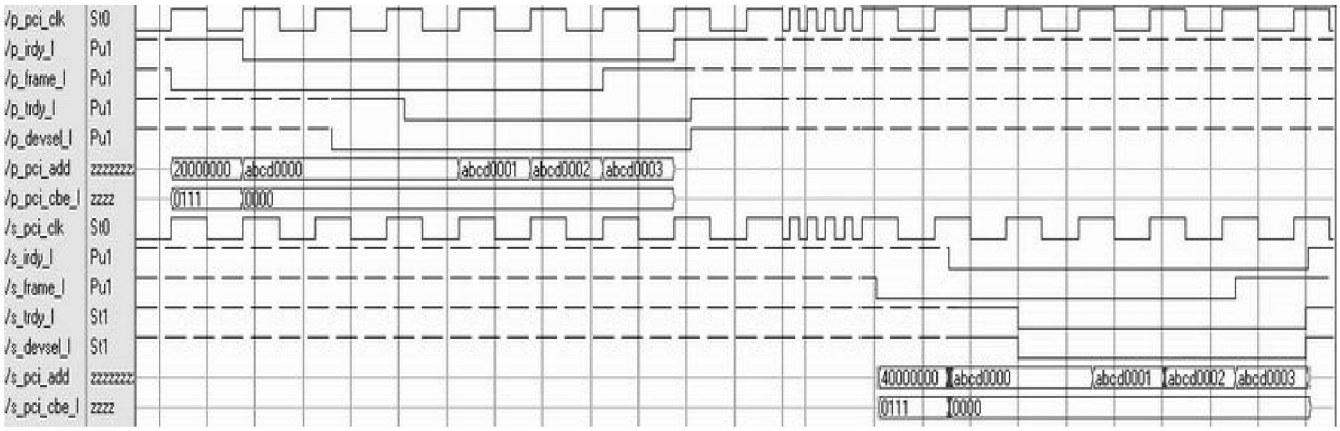

Figure 5. Write transaction

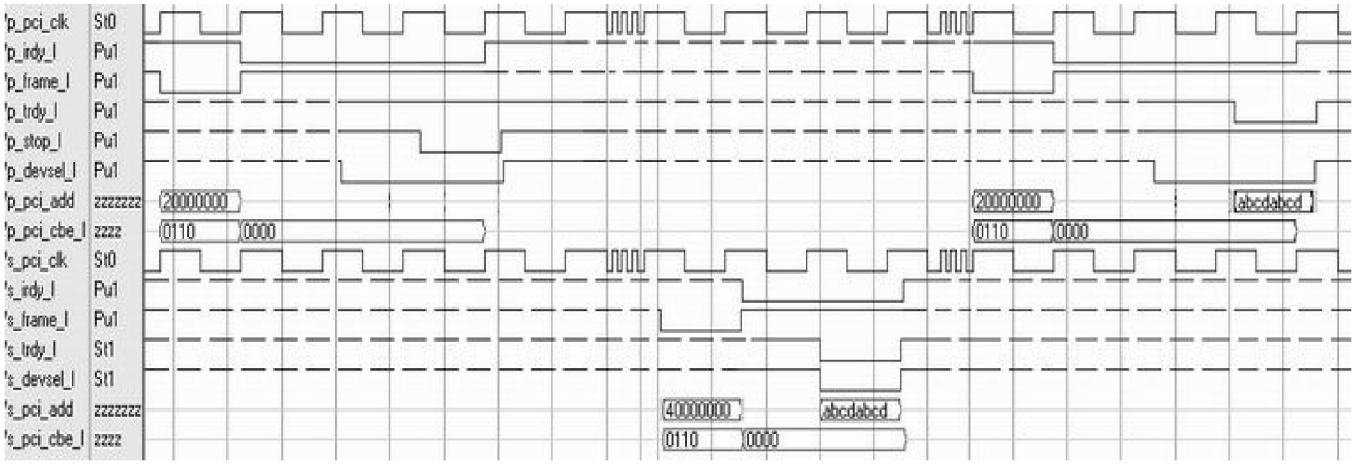

Figure 6. Read transaction

After accepting the data, abcd0000，abcd0001, abcd0002 and abcd0003, from the primary PCI bus, it originates a write transaction with the translated address, 40000000, to the target device on the secondary PCI bus. When the bridge accepts response from the target device, it sends data to the target device.

The read transaction is shown in Fig.6. The first eight signals belong to the primary interface of the bridge, while the others belong to the secondary interface. The bridge uses target retry to terminate the read transaction with address, 20000000, initiated by the master device on the primary PCI bus and originates a read transaction with the translated address, 40000000, to the target device on the secondary PCI bus, accepting the data, abcdabcd, from the target device. When the primary PCI bus retries the read transaction, the bridge responds it and sends the data, abcdabcd, to the primary PCI bus.

V. CONCLUSION

The design of the non-transparent PCI-to-PCI bridge based on Verilog HDL and ALTERA Cyclone FPGA is discussed. The bridge has passed the timing simulation and will be used in a 1553 intelligent communication board. The above timing waveforms of the PCI transactions show that the design of the bridge meets the requirements of the PCI specification.

REFERENCES

[1] Tom Shanley, Don Anderson, PCI System Architecture, Addison-Wesley Longman Inc, 1996.

[2] PCISIG, PCI Specification, Revision 2.2, 1998, http://www.pcisig.com/

[3] PCISIG, PCI-to-PCI Bridge Architecture Specification, Revision 1.2, 2001, http://www.pcisig.com/

[4] PICMG, CompactPCI Specification, Revision 3.0, 1999, http://www.picmg.com/

[5] Intel Inc., 21555 Non-Transparent PCI-to-PCI Bridge datasheet, 2004, http://www.intel.com/design/bridge/21555.htm

[6] Altera Inc., Cyclone Device Handbook, 2004, http://www.altera.com/products/devices/cyclone/cyc-index.jsp