Design of network adapter compatible OCP for high-throughput NOC

Wang Jian¹, a, Yang Zhijia¹, b

¹Lab of Industrial Control Network and System
Shenyang Institute of Automation, CAS
Shenyang, China

a wangjian710@sia.cn, b yang@sia.ac.cn

Keywords: network on chip; adapter; OCP interface protocol; burst transmission; virtual channel

Abstract. Network adapter is one of the most important components of NOC which is recommended to be used in present and future SOCs. In this paper, a network adapter compatible with OCP interface protocol is designed to enable the integration of IP cores from different providers into an on-chip interconnection network. In order to trade off communication performance with complication, the design of network adapter is based on a single global address space and supports burst transmission and virtual channel technology. With the goal of reducing transport latency, the processes of encapsulation and decapsulation are executing simultaneously with packet transmitting and receiving. Considering the simulation result, the network adapter presented in this paper gets satisfactory performance.

Introduction

NOC has been introduced as a new interconnection paradigm able to solve the increasing traffic and bandwidth demands for future systems due to its scalability, reusability and performance. It is composed of a set of network adapters, routers and links interconnected to each other. The network adapter, which IP core is attached to, decouples the core from the network and provides services at the transport layer in the ISO-OSI reference model. It handles the encapsulation and decapsulation of messages, and makes communication services transparently available with a minimal effort from the core.

Two characteristics must be met to implement the network adapter. The first is using standard interface protocol to fulfill the design reusability requirement and reduce system design time. The second is adopting an effective protocol conversion mechanism for better exploring the NOCs resources and performance to provide high-level communication services to the core by utilizing primitive services provided by the network hardware. In this paper, a network adapter compatible with OCP interface protocol is designed as a bridge between an IP core and the NOC switching fabric. In order to gain better communication performance with acceptable cost, it encapsulates read/write transactions into a single packet to transmit and uses wormhole switching technique to reduce transmission delay. The proposed network adapter also supports virtual channel and provides both best-effort services (BE) and guaranteed services (GS), and can interface to a core with both master and slave functionalities.

The remainder of this paper is organized as follows. The following section takes a brief look at related study by other research groups. Section 3 presents our network adapter’s basic functionality. Section 4 describes the architecture and detail implement of our network adapter. The simulation and synthesis results are given in Section 5 and finally a conclusion is made.

Related work

The network adapter’s function, general structure and service are outlined in [4]. The whole network adapter is divided into a front-end and a back-end sub-module. The front-end module implements a standardized interface protocol allowing core reuse across several platforms, and the back-end module performs data encapsulation and routing-related functions.
Radulescu et al. propose a network adapter offering guaranteed services, shared-memory abstraction, and flexible network configuration for the Æthereal\cite{5}. Stergiou et al. develop a highly parameterizable, synthesizeable network adapter for a SystemC library of the Xpipes NOC\cite{6}. The network adapter is split in two sub-modules, one for the request and the other for the response channel. Bhojwani and Mahapatra present a network adapter to interface OCP-compliant cores\cite{7}. Besides the basic functionality of packaging communication requests and responses, the network adapter provides additional services critical to communication in complex SOC.

The network adapters mentioned above all employ the NUMA (non-uniform memory access) communication model. In \cite{8}, it is proposed a network adapter to address the NORMA (no remote memory access) communication model. It employs OCP protocol and provides much flexibility of the modes for exchanging messages between the IP cores. But the network adapter may cost more resources because it must explicitly analyze the packets received from the network additionally.

**Network adapter functionality**

The network adapter is designed to comply with version 2.2 of the OCP specifications and contains both a master and a slave OCP interfaces. It handles all type of network traffic for both input and output directions, and can present both master and slave OCP signals at its OCP interface to the attached core, as shown in Fig. 1.

![Figure 1 Connectivity between IP core, NA and NOC communication infrastructure](image)

It is the responsibility of the network adapter to (i) extract the contents from the OCP interface, prepare the packets and dispatch them to the network and (ii) receive the packets from the network and translate them into the OCP interface signals. In addition, it is also able to handle end-to-end flow control, map a distributed shared address space to the NOC address space, and establish or withdraw a connection for GS service.

**Communication protocol translation.** Communication protocol translation is focus on address conversion and services mapping between both sides of IP core and on-chip interconnection network. All the IP cores connected to NOC are mapped to a single address space whose capacity lies on the width of address signal in OCP interface. In this design, the width of address signal is 32 bits, so the global address space is 4G bytes. The most significant 8 bits of address signal are defined as node address, used to identify the IP core, while the remaining 24 bits are defined as local address for each IP core's internal addressing. From the perspective of an IP core, the global address space is divided into two parts. One part is assigned to the network adapter connected to the IP core with the most significant 8 bits of address signal equal to its own node address; another part is used to access other IP cores on the chip with the most significant 8 bits of address addressing IP cores and the remaining 24 bits specifying the memory location inside the addressed IP core.

The network adapter provides IP cores with four types of services that are read, write, GS connection build and GS connection teardown. These services are performed in the OCP read and write transactions by encoding OCP interface signals as shown in Table 1.
Table 1. Mapping between services and OCP transactions

| Services               | OCP transactions                                      |
|------------------------|-------------------------------------------------------|
| Read                   | Read transaction with MAddr[31:24] != node address    |
| Write                  | Write transaction with MAddr[31:24] != node address   |
| GS connection build    | Write transaction with MAddr[31:24] = node address; MAddr[23] = 1; MData[9:8] != 0; MData[7:0] = destination node address |
| GS connection teardown | Write transaction with MAddr[31:24] = node address; MAddr[23] = 1; MData[11:10] != 0; MData[7:0] = destination node address |

**End-to-end flow control.** In order to ensure that all operations from the same IP core are executed in accordance with the order in which they are issued, the network adapter offers end-to-end flow control to IP core. If a network adapter has sent a request packet, it should be waiting the arrival of corresponding response packet before sending another request. Similarly, when the network adapter receives a request packet, it will transport relevant command to the IP core and wait for response. During this period, it will not receive and deal with new packets. Under this flow control mechanism, it’s sure that in each direction only one transaction is being handled at the same time.

**Encapsulation and decapsulation.** Encapsulation and decapsulation are independent of each other. When the network adapter detects a request or response transaction on OCP interface, it first encodes interface signals to generate a packet, and then transmits the packet across the network as a stream of flits. If a packet is received from the network, the network adapter analyzes and extracts relevant information from the packet, then generates OCP interface signals passed to the core.

![Figure 2. Data structure of packet](image)

The data structure of packet is shown in Fig. 2. Each packet is composed of header flit and payload flit, and each flit has a 32-bit data field and a 2-bit flag field. One bit of the flag field named sop indicates the start of the packet and another bit named eop indicates the end of the packet. As shown in Fig. 2, different types of operations are encapsulated into different frames with different data structure. The meaning of each data fields of packet is given in Table 2.

**GS connection build and teardown.** The network adapter supports guaranteed services which are implemented by sending GS packets in connection oriented mechanisms. So it needs to establish a dedicated connection prior to GS packets transport and withdraw the path after all GS packets have been transmitted.

There is a connection-mapping table stored in the network adapter. At the time of connection building, a virtual channel is allocated and related information is filled into the connection-mapping table; while during the time of connection teardown, a virtual channel is released and related information is deleted from the connection-mapping table.
### Table 2: Meaning of each data fields of packet

| Name       | Bits | Description                                              |
|------------|------|----------------------------------------------------------|
| dest_addr  | 8    | Destination address of packet                            |
| src_addr   | 8    | Source address of packet                                 |
| req        | 1    | Flag of request packet                                   |
| cmd        | 3    | Type of operation                                         |
| GS_BE      | 1    | Flag of GS packet                                         |
| GS_type    | 2    | Type of GS build or teardown operation                    |
| Del_sd     | 2    | Result of GS build or teardown operation                  |
| GS_VC_id   | 8    | Virtual channel used in build or teardown operation       |
| resp       | 2    | Information of response                                   |
| burst_length | 6  | Number of transmitting data                               |
| burst_type | 2    | Type of burst transmission                                |

### Architecture and implementation

An architectural sketch of the proposed network adapter is shown in Fig. 3 and highlights the major components.

Two complementary OCP interfaces are used to connect to IP core. Each interface has a special set of OCP signals shown in Table 3. The signals have the same definitions in both OCP interfaces but in opposite direction. The Data handshake and transfer pipelining are not supported at this moment, but may be considered for future development. Three types of burst operations, which are INCR, STRM and WRAP, are provided. Each burst transmission with up to 64 OCP words is encapsulated in a single packet to transport.

The slave interface control module, which acts as an OCP slave, has responsibility to receive command requests from IP core and return response information. It first encodes address, command and data information obtained from IP core and sends them to the packetizer module; then gets data and response information abstracted from received response packet by the depacketizer module and represents them as response.

The master interface control module works as an OCP master. It receives command, address and data information resolved from request packet through the depacketizer module and issues OCP request operations to IP core. After the request operations are executed by IP core, the master interface control module encodes data and response information received from IP core and passes them to the packetizer module.

### Table 3: Signal set of OCP interface

| Name         | Bits | Description                  |
|--------------|------|------------------------------|
| MAddr        | 32   | Transfer address             |
| MCmd         | 3    | Transfer command             |
| MData        | 32   | Write data                   |
| MRespAccept  | 1    | Master accepts response      |
| SCmdAccept   | 1    | Slave accepts transfer       |
| SData        | 32   | Read data                    |
| SResp        | 2    | Transfer response            |
| MBurstLength | 6    | Burst length                 |
| MBurstSeq    | 2    | Address sequence of burst    |
| MReqLast     | 1    | Last request in burst        |
| SRespLast    | 1    | Last response in burst       |
The packetizer module prepares flits sent out over the NOC. It packetizes encoded data and sets the fields of flits. Address, command and data information coming from the slave interface control module are encapsulated into request packet, while data and response information from the master interface control module are encapsulated into response packet.

The depacketizer module accepts flits from the NOC and extracts relevant information from each field of flits. According the type of received packet is whether request or not, these information are forwarded to the master interface control module or the slave interface control module.

The VC manager module has two basic functions. First it monitors the use of virtual channels in both input and output links and stores the state information into VC allocation table. By looking up the VC allocation table, the VC manager module allocates virtual channel for a packet which is going be sent. Second it establishes and withdraws GS connections for GS packets, then records related information into the connection mapping table.

The link control module provides an interface for the network adapter to the NOC. It adopts handshake mechanism to control the link access and avoids overflow occurs during packets transmission.

---

**Experimental results**

**Performance evaluation.** An RTL NOC simulator is implemented to assess the efficiency of the proposed method. The simulator models all major components of the NOC such as network adapters, routers, and links. The router has a typical state-of-the-art structure including input buffers, a virtual channel allocator, a routing unit, a switch allocator and a crossbar. It adopts XY routing and wormhole switching. By specified configuration parameters such as number of virtual channels, link width and buffer depth for the simulator, it’s able to emulate different systems.

A test case with 25-node (5×5) 2D mesh on-chip network is implemented. Each node is considered to have a processor and a memory (master and slave cores with network adapter), and sends data to other nodes with uniform probability periodically. We keep the amount of data sent by each node constant and increase burst size in every simulation. As the performance metric, we use latency defined as the number of cycles between the initiation of a request operation issued by the master and the time when the response is completely accepted and responded by the slave. The request throughput is defined as the number of the successful read/write request injections into the network per unit time. The execution time is defined as the time to complete all operations. The result is shown in Table 4. Obviously, increasing burst size of OCP operation is able to improve the communication efficiency of on-chip interconnection network.

| Burst size | Latency | Request throughput | Execution time[ns] |
|------------|---------|--------------------|--------------------|
| 2          | 72      | 0.237              | 1,081,683          |
| 4          | 77      | 0.465              | 551,993            |
| 8          | 90      | 0.897              | 287,253            |
| 16         | 120     | 1.673              | 155,053            |
| 32         | 179     | 2.950              | 89,613             |
| 64         | 391     | 4.770              | 59,423             |

**Synthesis report.** The network adapter is implemented on an Altera StratixII FPGA. The balanced optimization technique is used in synthesis to balance high performance with minimal logic usage. Logic utilization and max operating clock frequency of the network adapter are under the influence of the number of virtual channels provided. As the virtual channels increase, logic utilization rises and operating clock frequency reduces. The reason for this trend is that the amount of virtual channels determines the size of VC allocation table and connection mapping table and affects how complicated the combinational logic is in VC manage module. As a typical example, the network adapter with 4 virtual channels costs 1741 logic elements and is able to run in 340 Mhz.
Conclusion

In this paper, we presented an on-chip network adapter compatible with OCP interface protocol which supports burst transmission and virtual channel technology for the purpose of high request throughput. The proposed network adapter has been implemented on an Altera StratixII FPGA programmed with the Verilog language. An RTL NOC simulator was used to evaluate the efficiency of the proposed method and the simulation result showed that the network adapter gets satisfactory performance.

Acknowledgment

The authors would like to give sincere gratitude for the workmates in the SOC team with their significant and constructive comments and suggestions. The work is supported by the "Strategic Priority Research Program" of the Chinese Academy of Sciences (Grant No. XDA06020402) and the National High Technology Research and Development Program of China (863 Program) (No. 2011AA040102).

References

[1] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg and D. Lindqvist, “Network on Chip: An Architecture for Billion Transistor Era”, in Proc. of the IEEE NorChip Conference, (2000), p.166

[2] W. Dally and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks”, in Proc. of DAC, (2001), p.684

[3] L. Benini and G. de Micheli, “Networks on chips: A new SoC paradigm”, IEEE Computer, vol.35, Issue 1, (2002), p.70

[4] L. Benini and G. De Micheli, Networks on chips: technology and tools, Morgan Kaufmann Publishers, (2006), p.213

[5] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema and P. Wielage, “An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration”, Proceedings of the Design, Automation and Test in Europe, vol.2, (2004), p.878

[6] S.Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi and G. de Micheli, “Xpipes Lite: A Synthesis Oriented Design Library For Networks on Chips”, Proceedings of the conference on Design, Automation and Test in Europe, vol.2, (2005), p.1188

[7] P. Bhojwani and R. N. Mahapatra, “Core Network Interface Architecture and Latency Constrained On-Chip Communication”, Proceedings of the 7th International Symposium on Quality Electronic Design, (2006), p.358

[8] E. Carara, A. Mello and F. Moraes, “Communication Models in Networks-on-Chip”, Proceedings of the 18th IEEE/IFIP International Workshop on Rapid System Prototyping, (2007), pp.57