International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:4, No:9, 2010

An Address-Oriented Transmit Mechanism for GALS NoC

Yuanyuan Zhang, Guang Sun, Li Su, Depeng Jin and Lieguang Zeng

***Abstract*—**Since Network-on-Chip (NoC) uses network interfaces (NIs) to improve the design productivity, by now, there have been a few papers addressing the design and implementation of a NI module. However, none of them considered the difference of address encoding methods between NoC and the traditional bus-shared architecture. On the basis of this difference, in the paper, we introduce a transmit mechanism to solve such a problem for global asynchronous locally synchronous (GALS) NoC. Furthermore, we give the concrete implementation of the NI module in this transmit mechanism. Finally, we evaluate its performance and area overhead by a VHDL-based cycle-accurate RTL model and simulation results confirm the validity of this address-oriented transmit mechanism.

***Keywords*—**Network-on-Chip, Network Interface, Open Core Protocol, Address.

I. Introduction

THE traditional bus-shared architecture tends to cause the bottleneck effect in the high-performance SoC(System on Chip). To solve this problem, NoC (Network on Chip) was proposed as a new interconnection architecture [1], [2], [3]. NoC is comprised of three fundamental components: network interfaces (NIs), routing nodes and links, respectively. Network interfaces implement the interface by which IP cores connect to the NoC. Their function is to decouple computation from communication. In this way, designs of IP cores and NoC can be made separately, and consequently, the design productivity can be improved. Until now, a few papers have addressed problems particular to the design of a NI module [4], [5]. But only [4] presented a NI structure in detail. The paper didn't consider the difference of address encoding methods between NoC and the traditional Bus-shared architecture. To emphasize this difference, this paper proposes a transmit mechanism in order to implement the better compatibility with the IP cores which are designed for the bus-shared architecture. Since the Open core Protocol (OCP) delivers an only non-proprietary, openly licensed, core-centric protocol which comprehensively describes the system-level integration requirements of IP cores, we consider the condition in which OCP is used in all IP cores.

The rest of the paper is organized as follows. In Section II, we introduce the principle of the address-oriented transmit mechanism. Then, in Section III, we show the structure of the corresponding NI and the address converter. In Section IV, we give the simulation results to confirm the validity of our mechanism. Finally, we conclude the work in Section V.

The authors are all with State Key Laboratory on Microwave and Digital Communications, Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
E-mail: zhangyuanyuan07@mails.tsinghua.edu.cn.
This work is supported by the National High Technology Research and Development Program with No.2008AA01Z107

II. The Address-Oriented Transmit Mechanism

In the bus-shared architecture, each address corresponds to the memory space uniquely, namely, different memories have different addresses, called memory address in this paper whereas, in NoC, there are two parts of each address. One part is the address of IP cores in NoC, called network address, and another part is the address of the memory space of the corresponding IP core, namely memory address. Thus, we need a mechanism to implement the address transformation between the traditional bus-shared architecture and NoC.

OCP defines a point-to-point interface between two communicating entities. One entity acts as the master of the OCP instance, and the other as the slave. Note that only the master can present commands and is the controlling entity. The slave responds to commands presented to it, either by receiving data from the master, or presenting data to the master. So for two entities to communicate in a peer-to-peer fashion, there need to be two instances of the OCP connecting them - one where the first entity is a master, and one where the first entity is a slave [6]. Hence, by using OCP, NoC connects IP cores in the way, as shown in Fig. 1.

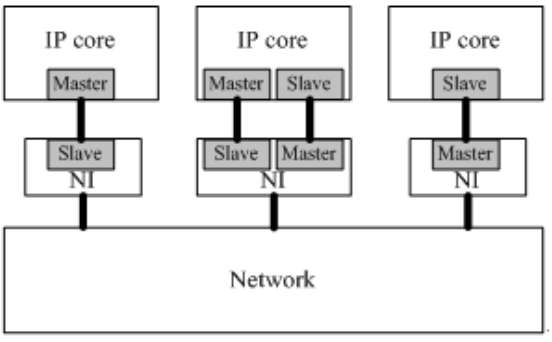
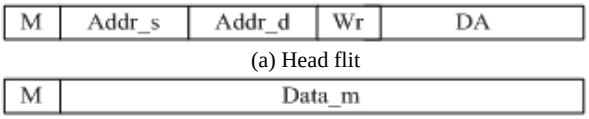


Fig. 1 The interconnection architecture by using OCP

As shown in Fig. 1, the characteristics of IP cores determine whether the core needs master, slave, or both of them. And a NI must have the module which acts as the complementary side of the OCP for each connected entity.

In this paper, we only consider the IP cores which represent

CPUs or memories. The request signals are stored in NIs in flit form which is the minimum flow control digit. Moreover, each packet is comprised of flits. Specifically, there are three kinds of flit, respectively head flit, data flit and tail flit. The head flit indicates the start of a new packet, the data flit indicates the continuation of a packet and the tail flit indicates the end of a packet. They consist of the subfields, as shown in Fig. 2.



(a) Head flit

(b) Data or tail flit

Fig. 2 The formation of flits

As shown in Fig. 2, M indicates the kind of flits. It uses two bits to mark them. In particular, "00" indicates the head flit, "01" indicates the data flit and "10" indicates the tail flit. Addr_s indicates the source IP core's network address; Addr_d indicates the destination IP core's network address; Wr indicates the two operations, i.e., read or write; DA indicates the memory address to operate in the destination IP core; Data_m indicates the read or written data.

In the paper, we only consider the cases in which only simple write and read operations are executed. When a processor needs to write or read some data with the memories, the corresponding NI must convert the address from IP cores to the address required by NoC. One feasible way is to store the necessary information in each NI but this approach needs lots of memory spaces and leads to much area overhead. Thus, instead of this method, we store all the necessary information in the address converter, and we employ the transit mechanism in which, when a processor needs to write or read some data with the memories, the corresponding NI sends the memory address to the address converter at first. Then, on the basis of the original memory address, the address converter finds the corresponding network address, and sends it back to the NI. Next, the NI adds the network address to the head flit, and then executes the write or read operation between the corresponding IP cores. When a CPU wants to take a read operation, due to the characteristic of OCP, the corresponding NI must wait, and can't execute any other operation until the CPU receives the data that aim to read.

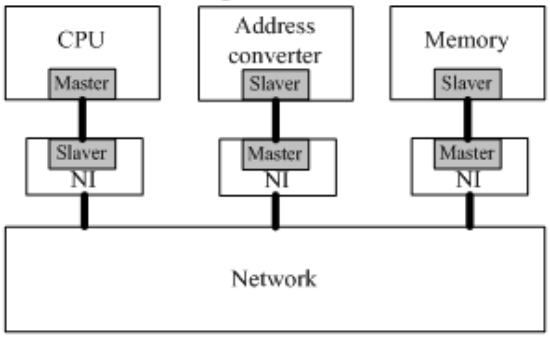
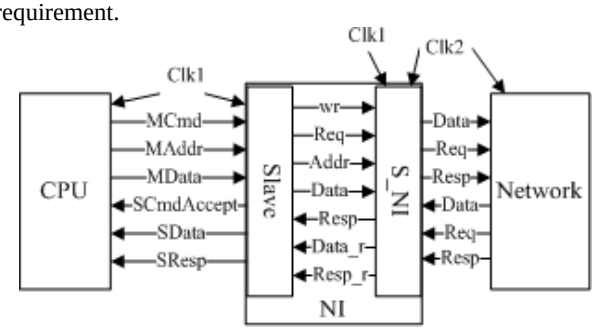


Fig. 3 The interconnection architecture in this paper

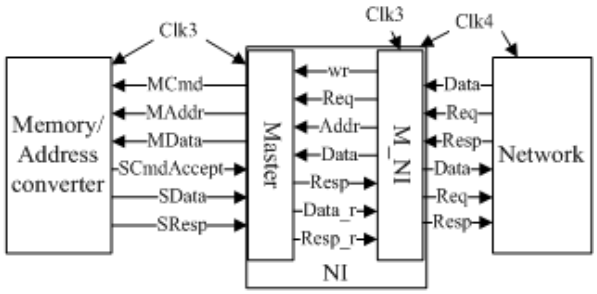
From the description of the mechanism, it is found that an address converter interface only requires an OCP slave. From Ref. [6], we can conclude that a CPU interface usually only requires an OCP master, and a memory interface usually only requires an OCP slave, so the IP cores in this paper are interconnected by the way shown in Fig. 3.

III. The Structure

According to the transmit mechanism, the structure of NI is shown in Fig. 4. We use asynchronous FIFOs in NIs for the global asynchronous locally synchronous (GALS) NoC's requirement.



(a) The way that NI connects to CPU



(b) The way that NI connects to memory/address converter

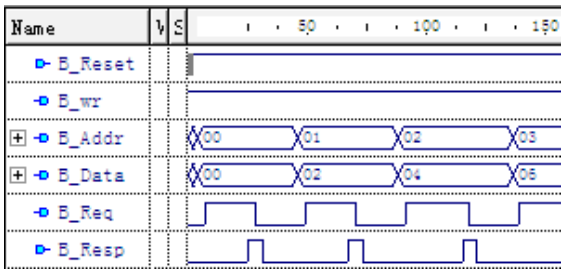
Fig. 4 The NI structure

The signals used in Fig. 4 are explained in detail as below:
(i) The signals between IP cores and NIs:
MCmd: Transfer command, indicating write/read operation,
MAddr: Destination IP core's memory address,
MData: Data to write,
SCmdAccept: Slave accepts a transfer,
SData: Data to read,
SResp: Transfer response,
(ii) The signals in NIs:
Wr: The type of operation, including write and read,
Req: Transfer requirement,
Addr: Memory address, the same with MAddr,
Data: Data to write,
Resp: Transfer response,
Data_r: Data to read,
Resp_r: Read response,
The signals between network and NIs:
Data: Transfer data,
Req: Transfer requirement,
Resp: Transfer response.
The address converter has the same interface as memories

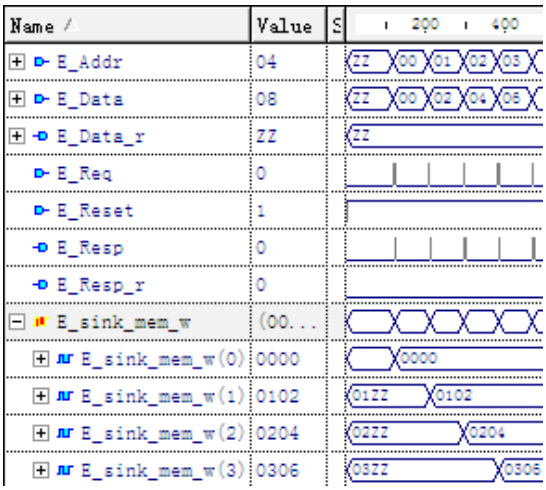
and it stores the mapping between memory addresses and network addresses.

IV. Simulation Results

We use a VHDL-based cycle-accurate RTL model to evaluate the performance of the proposed mechanism. For simplicity, we connect a CPU, a memory and an address converter together by a routing node simply. The routing node implements the interconnection among them, and routes messages according to their network address.



(a) Signals at the source IP core



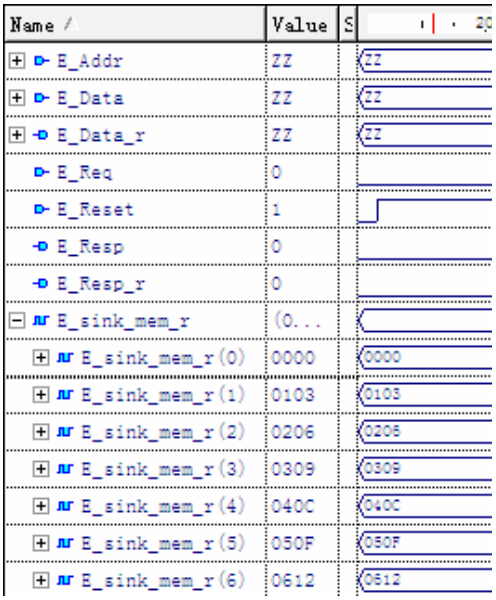
(b) Signals at the destination IP core

Fig. 5 Write operation

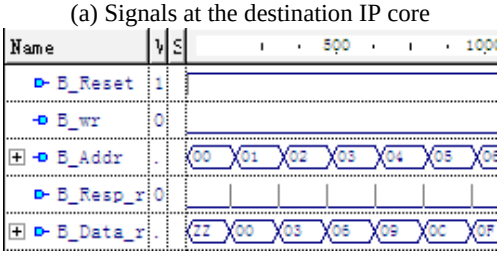
As shown in Fig. 5, the source IP core executes a write operation, and from Fig. 5 (b), it is observed that the destination IP core receives the data sent to the corresponding memory space correctly.

As shown in Fig. 6, the source IP core executes a read operation, and from the figure, it is observed that the source IP core receives the read data from the corresponding memory space correctly.

We evaluate the performance by operation time that refers to the time interval between the moment when the source IP core sends a requirement and the moment when the corresponding IP core receives the corresponding data. From the simulations, it is found that, without blocking, a write operation occupies 39 clock cycles and a read operation occupies 61 clock cycles.



(a) Signals at the destination IP core



(b) Signals at the source IP core

Fig. 6 Read operation

We use the device Stratix EP1S80F1508I7 on the Quartus II platform to synthesize them. It is found that, if the flit is 19 bit width and the FIFOs in NIs can store five flits at most, then the NI occupies 867 logic unites for CPUs and meanwhile 802 logic units for memories and address converters.

V. Conclusion

In this paper, we proposed an address-oriented transmit mechanism to implement the NoC's compatibility with the traditional IP cores used in the traditional bus-shared architecture. The structure of NIs and the address converter in this mechanism was well designed and finally, its validity is confirmed via the evaluation of the performance and area overhead by a VHDL-based cycle-accurate model.

References

[1] W.J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in Proceeding of the 38th Design Automation Conference (DAC01), vol.1, pp.684-689, 2001.

[2] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," IEEE Trans. Comput.., vol 54, no. 8, pp. 1025-1040, 2005.

[3] R. Marculescu, U.Y. Ogras, L.S. Peh, N.E. Jerger and Y. Hoskote, "Keynote Paper Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," IEEE Trans on compuer-aided design of integrated circuits and systems, vol. 28, no. 1, pp. 3-21, 2009.

[4] 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," in Proceeding of the 2004 Design, Automation and Test in Europe Conference (DATE04), 2004.
[5] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø, An OCP compliant network adapter for GALS-based SoC design using MANGO network-on-chip," in Proceedings of International Symposium on System-on-Chip (SOC05), pp, 171-174, 2005.
[6] http://www.ocpip.org/