Network interface design based on mutual interface definition

Bingjie Xia and Kejun Wu

Department of Information Science & Electronic Engineering,
Zhejiang University,
Hangzhou 310027, China
E-mail: icysummer@zju.edu.cn E-mail: wkj_iseer@126.com

Chunchang Xiang

NVIDIA Research Center,
Building 9, Zhangjiang Hi-Tech Park,
No. 399, Keyuan Road, Shanghai, 201203, China
E-mail: jxiang@nvidia.com

Mei Yang

Department of Electrical and Computer Engineering,
University of Nevada,
4505 Maryland Parkway, Box 454026, Las Vegas, NV 89119, USA
E-mail: mei.yang@unlv.edu

Peng Liu* and Qingdong Yao

Department of Information Science & Electronic Engineering,
Zhejiang University,
Hangzhou 310027, China
E-mail: liupeng@zju.edu.cn E-mail: yaoqd@zju.edu.cn
*Corresponding author

Abstract: This paper proposes a novel network interface design method, referred to as mutual interface definition based method. It decouples resource dependent part (RDP) from resource independent part (RIP) by mutual interface definition. These two parts can be designed independently, thus the design flexibility and reusability of network interface can be enhanced. Moreover, a network interface component library consisting of multiple RDP and RIP components is proposed to be built. The networks-on-chip designers can choose appropriate components from the library to construct network interface design. From the perspective of RDP, the network interface achieves backward compatibility with the existing protocols such as AMBA AHB and OCP. From the perspective of RIP, the network interface provides a configurable structure supporting multicast transfer and adaptive routing algorithm extensions. The proposed network interface designs are implemented in TSMC 90-nm CMOS standard cell technology and can work at the frequency of 1.12 GHz to 1.35 GHz.

Keywords: networks-on-chip; NoC; network interface; NI; resource dependent part; RDP; resource independent part; RIP.

Reference to this paper should be made as follows: Xia, B., Wu, K., Xiang, C., Yang, M., Liu, P. and Yao, Q. (2010) 'Network interface design based on mutual interface definition', *Int. J. High Performance Systems Architecture*, Vol. 2, Nos. 3/4, pp.168–176.

Biographical notes: Bingjie Xia is a PhD student in Communication and Electronic Engineering from Zhejiang University, China. Her research focuses on high performance digital VLSI design, embedded processor and many-core on-chip interconnection networks.

Kejun Wu is a PhD student in Communication and Electronic Engineering from Zhejiang University, China. His research interests include parallel computer architecture and VLSI design.

Chunchang Xiang received his BEng in Communication and Electronic Engineering from Zhejiang University, China, in 2006. His research focuses on multi-processor architecture design and on-chip interconnection networks.

Mei Yang received her PhD in Computer Science from the University of Texas at Dallas. She is an Assistant Professor in the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas. Her research focuses on computer engineering, computer architectures, embedded systems and networks-on-chip.

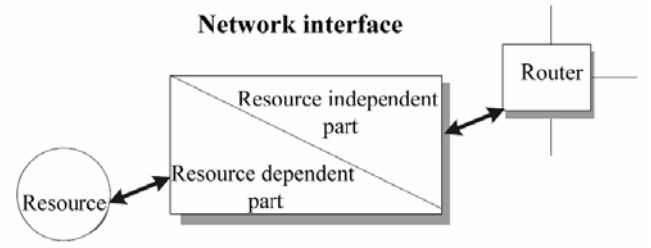
Peng Liu received his PhD in Communication and Electronic Engineering from Zhejiang University, China. He is an Associate Professor in the Department of Information Science and Electronic Engineering, Zhejiang University. His research focuses on embedded processor, multiprocessor system-on-chip architectures, on-chip interconnection networks and real-time operating system.

Qingdong Yao is a Professor in the Department of Information Science and Electronic Engineering, Zhejiang University, China. His research interests include multi-core system-on-chip, on-chip interconnection networks, high performance CPU and DSP design, communication and video codec.

1 Introduction

With the growing complexity in consumer embedded products, new tendencies forecast multi-processor system-on-chip (MPSOC) consisting of complex integrated components. Intercommunication requirements of MPSOC will not be feasible using a single shared bus or a hierarchy buses due to their poor scalability. Networks-on-chips (NoCs) (Benini and Micheli, 2002; Atienza et al., 2008) have been proposed as a promising replacement. Network interface (NI), as one of the important components of NoC, attracts more attention from designers.

Figure 1 NI structure



NI in NoC is responsible for adapting the IP cores to the on-chip networks. It is always denoted as glue logic necessary to enable a resource to communicate with the networks. Both the characteristics of IP cores and NoCs should be taken into considerations simultaneously when designing a NI, which makes the NI design complicated. Based on this, an idea occurred in our mind. Can we find one NI design method to decouple the IP core dependent part and NoC dependent part? If these two parts could be implemented separately, the design flexibility will be enhanced. Jantsch and Tenhunen (2003) have proposed that the internal design of a NI is composed of two parts as shown in Figure 1. The part connected to the IP cores is resource dependent part (RDP) and the other part is resource independent part (RIP). However the NI partition is just on the conception stage. This paper proposes a new NI design approach based upon this representation, referred to as NI based on mutual interface definition (NIMID). It decouples the RDP and RIP design by a set of clear mutual interface signals definition and explicit function assignment to these two parts. To the best of our knowledge, the NIMID is the first to address a NI design by decoupling RDP and RIP. Moreover, the reusability of NI is facilitated by the decoupling. The RDP attached to the same resource element interface can be reused, and the RIP component attached to the NoC with the same characteristics can be reused as well.

The key contributions of this work include:

1 It proposes an innovative NI design method called NI based on the mutual interface definition, which implements decouple of RDP and RIP. Then the RDP and RIP can be designed independently without considering the characteristics of the other component to meet the goal of design flexibility improvement.

2 It proposes to build a NI component library which consisting of multiple RDP and RIP components. Designers can choose optimum components from the library according to the requirement, and then construct their NI through the mutual interface definition.

The rest of this paper is organised as follows. In Section 2, some related works about NI design are introduced. In Section 3, a design flow based on the NIMID is given. In Section 4, the mutual interface definition is declared and the structures of RDP and RIP are illustrated as well. In Section 5, the construction of the NI component library is discussed. Section 6 provides the performance evaluation of proposed NIMID. Section 7 gives the concluding remarks.

2 Related work

The NI for NoC, acting as the interface between the IP cores and routers, has received considerable attention from NoC engineers.

Æthereal (Goossens et al., 2005) offers a modular NI architecture (Micheli and Benini, 2006), which allows flexible instantiation. It uses a transaction-based protocol to achieve compatibility with existing bus protocols. However, the Æthereal NI does not take packets reordering problem into consideration, which cannot be used in NoC with adaptive routing algorithm. NI in MANGO (Bjerregaard et al., 2005) is designed as a bridge between an OCP interface

and the NoC fabric. It adopts message passing method to support NoC without clock. The Xpipes NI (Bertozzi and Benini, 2004; Dall'Osso et al., 2003) is also compliant with OCP. Two types of configurable NI components are implemented to support master and slave transfer functions, named initiator (attached to system masters) and target (attached to system slaves). The NI in Cell processor (Ainsworth and Pinkston, 2007a, 2007b) is responsible to transfer data from a direct memory access (DMA) to the interconnect network named element interconnect bus. It achieves a high frequency of 1.6 GHz.

Besides these designs, some other problems in NI designs are discussed, including low power, low latency, reordering mechanism issues, etc. (Ainsworth and Pinkston, 2007a, 2007b; Radulescr et al., 2005; Kundu and Chattopadhyay, 2007; Bhojwani and Mahapatra, 2006; Xu et al., 2007; Daewook et al., 2006). However, all these state-of-the-art NI designs have to consider both the characteristics of IP cores and NoC, which makes the NI design complicated. Although Jantsch and Tenhunen (2003) have proposed that the NI design can be partitioned into RDP and RIP, it does not mention how to decouple them. That is to say that no designs have talked about decouple between RDP and RIP.

3 NI design flow

This paper proposes a new NI design method based on the mutual interface definition, which implements decouple of RDP and RIP. Based on this method, a new NI design flow is proposed as shown in Figure 2. The design flow can be divided into four main steps as follows.

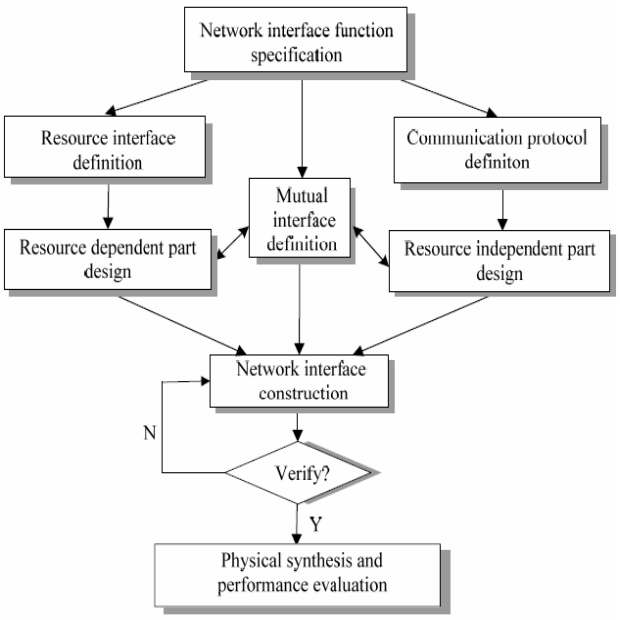
1 *Resource element interface definition and RDP design:* NI is responsible for adapting the resource elements to the on-chip networks. After the specification of NI function, three important definitions are raised, including resource element interface definition, network communication protocol definition and a proposed mutual interface definition. The first step is to determine the resource element interface definition and then the RDP is designed. The RDP design only considers the resource interface and the standard mutual interface, regardless of the network communication protocol. The main function of the RDP is to configure the communication control signals to adapt resource I/O pins to mutual interface. The RDP can be reused in the NI when connected to the same resource interface.

2 *Characteristics of NoC definition and RIP design:* Determine the characteristics of NoC, which includes communication mode, switching technique, flow control, routing techniques, etc. Then design the RIP based on the characteristics of NoC and mutual interface definition, regardless of the resource element interface. The function of RIP includes communication protocol implementation, packetisation and depacketisation of data, buffering and synchronisation of multi-clock domains.

3 *NI construction:* After the completion of RDP and RIP design, these two parts can assembled to construct a NI by the defined standard mutual interface.

4 *Back-end synthesis and performance evaluation:* After the function verification of NI, perform the back-end synthesis. And evaluate the NI performance to check whether the design specification is satisfied.

Figure 2 NI design flow



4 Mutual interface definition

The RDP is responsible for generating control interface signals according to the signals from IP core interface and coordinating the data signals between IP cores and RIP. The RIP, working as the most significant component, is responsible for multiple functions, including implementing network communication protocol, end-to-end flow control, data packetisation, depacketisation, multi-clock domain synchronisation, etc. The mutual interface signals between RDP and RIP are defined which is the key point of the NIMID design.

4.1 Mutual interface definition description

The mutual interface has been classified into two categories. One is the control interface and the other is data interface. The control interface involves signals that are used to control the NI communication transaction. The data interface involves the data signals transferred between RDP and RIP.

4.1.1 Control interface

The control interface is a collection of signals that are used to configure and initiate a communication transaction. The proposed NI supports peer-to-peer transfer and optional multicast and reordering mechanism. Besides the basic

direct peer-to-peer transfer, the proposed NI also supports indirect transfer. A direct peer-to-peer transfer is initiated by source node and communicates between source node and destination node. An indirect transfer is initiated by source node and communicates between intermediate node and destination node, which is useful for heterogeneous multiprocessor system. In order to support the above-mentioned transfer modes, the control interface signals consist of a set of required basic signals and a number of optional signals that can be configured to support additional network communication as shown in Figure 3.

Figure 3 Control interface signal declaration

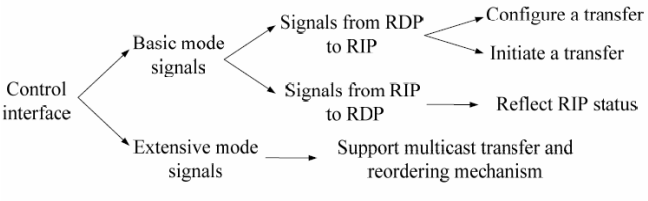


Table 1 Control interface signal

Signal name	*Width*	*Driver*	*Function*
Basic signals			
Start	1	RDP	Transfer start control signal
Mode	1	RDP	Transfer mode: 1 – direct, 0 – indirect
Write	1	RDP	Transfer direction: 1 – write, 0 – read;
Pkg_size	Cfg.	RDP	Size of a single packet
Pkg_count	Cfg.	RDP	Packet count
Dest_node	Cfg.	RDP	Destination node number
Dest_addr	Cfg.	RDP	Destination address
Src_node	Cfg.	RDP	Source node number
Src_addr	Cfg.	RDP	Direct transfer: source address Indirect transfer: intermediate address
Inter_node	Cfg.	RDP	Intermediate node for indirect transfer
Trans_ongoing	1	RIP	Transfer ongoing
Trans_done	1	RIP	Transfer completed
Trans_error	1	RIP	Transfer error
Adaptive routing algorithm extension			
Adaptive_en	1	RDP	Adaptive routing algorithm enable signal
Multicast extension			
Multi_en	1	RDP	Multicast enable signal
Multi_destnode	Cfg.	RDP	Multicast destination node
Multi_amount	Cfg.	RDP	Amount of multicast destinations
Multi_groupid	Cfg.	RDP	Multicast group identity

Note: Cfg. denotes configurable.

The basic mode signals can be divided into two types from the perspective of signal driving source. One type is sent from RDP to RIP, which is used to configure and initiate a transfer. In order to configure a transfer, the interface should be able to describe the transfer. A transfer can be simply described by the following parameters: source node number, source address, destination node number, destination address, packet width, packet count and transfer direction (read/write). For an indirect transfer, two more parameters about the intermediate nodes are needed as well: intermediate node number and intermediate address. After the transfer is described, a control signal is needed to start the transfer. The other type is sent from RIP to RDP, which is used to inform the RDP of the status of RIP. The status includes transfer ongoing, transfer error and transfer done. Besides, four additional signals are defined to support multicast transfer. The signal Multi_destnode is used to represent the destination nodes, which adopts the bit string encoding method (Culler et al., 1999). The signal Multi_amount is used to represent the amount of the destination nodes since the number of destination nodes is more than one for multicast transfer. The signal Multi_groupid is used to distinguish this transfer with others. The control interface signals are listed in Table 1.

4.1.2 Data interface

The data interface signals are used to transfer data between RDP and RIP as listed in Table 2. Besides the data signals, two groups of control signals are defined. RIP reads or writes the data according to the value of Data_out_valid and Data_in_done, regardless of the RDP status. It is similar for RDP. RDP reads or writes the data according to the value of Data_in_valid and Data_out_done, regardless of the RIP status.

Table 2 Data interface signal

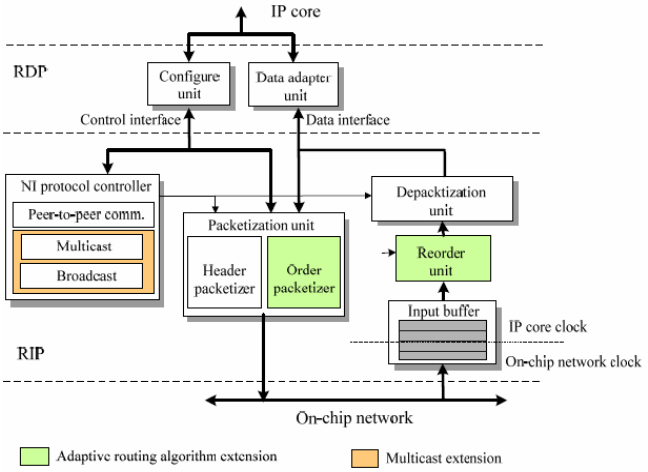
Signal name	*Width*	*Driver*	*Function*
Data_out	Cfg.	RDP	Data signal read from RIP
Data_out_valid	1	RDP	Data out status: 1 – valid, can be read out; 0 – invalid, cannot be read out
Data_out_done	1	RIP	Data out status: 1 – has finished reading out; 0 – has not finished reading out
Data_in	Cfg.	RIP	Data signal write to RDP
Data_in_valid	1	RIP	Data in status: 1 – valid, can be read in; 0 – invalid, cannot be read in
Data_in_done	1	RDP	Data in status: 1 – has finished reading in; 0 – has not finished reading in
Wen	1	RIP	Write enable signal

Note: Cfg. denotes configurable.

4.2 NI construction

The architecture of NI design based on the mutual interface definition is shown in Figure 4. It is composed of RDP and RIP, which are interconnected through the mutual interface.

Figure 4 Architecture of NIMID (see online version for colours)



4.2.1 RDP design

The RDP consists of two units: a configure unit and a data adapter unit. The configure unit is used to configure the control interface signals listed in Table 1. The efficiency of configure process depends on the scale of network and resource element interface. Take a network with 16 nodes as an example, four 32-bit registers need to be configured to initiate a transfer, including destination address register, source address register and two control registers. Besides the destination address and source address, other signals are stored in two control registers as given in Table 3.

The configuration steps of the NIMID are given as follows:

1 write the destination address

2 write the source address

3 write the control register 1

4 write the control register 2.

If the interface of IP cores supports data with the width of 32 bits, three cycles are needed for a basic direct peer-to-peer transfer. Four cycles are needed for indirect peer-to-peer and multicast transfers. If the interface can support data with wider width, the configure process can be completed with fewer cycles to reduce the latency of transfer initiation. Besides these four 32-bit registers, there is still a status register that is used to inform the IP cores of the status of NI.

The data adapter unit is designed to adapt data signals compliant with IP cores to those compliant with data interface. Assume the data width of IP cores is 32-bit, while the data width of data interface is 64-bit. The function of the data adapter is to packet two data from IP cores to constitute a 64-bit data for data interface.

Table 3 Control registers description

Bit	*Signal name*	*Description*
Control register 1		
[31:16]	Count	Transfer data count
[15:12]	Dest_node	Destination node
[11:8]	Src_node	Source node
[7]	Start	Start control signal
[6]	Mode	Transfer mode
[5]	Write	Write enable signal
[4:3]	Size	Packet size
[2]	Rsvd	Reserve bits
[1]	Adaptive_en	Adaptive enable control signal
[0]	Multi_en	Multicast enable control signal
Control register 2		
[31:28]	Rsvd	Reserved bits
[27:24]	Inter_node	Intermediate node
[23:8]	Multi_destnode	Destination node for multicast
[7:4]	Multi_amount	Transfer count for multicast
[3:0]	Multi_groupid	Group id for multicast

4.2.2 RIP design

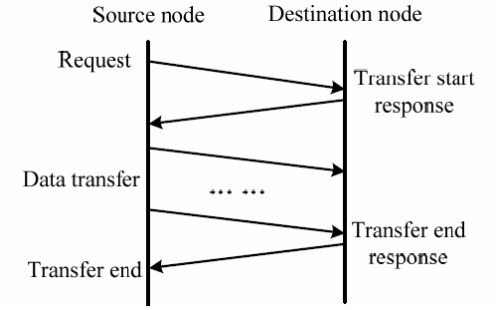
The RIP adopts a configurable structure to provide deployment flexibility based on the signals of control interface. The RIP takes care of multiple functions, including network communication protocol implementation, end-to-end flow control, data packetisation, depacketisation, multi-clock domains synchronisation, etc. The RIP is composed of five main components: a protocol controller unit, a packetisation unit, a depacketisation unit, an input buffer and output buffer as shown in Figure 4.

The protocol controller unit acts as the main controller in RIP, which implements the communication protocol and controls the operation of other units. Three transfer modes that the NI supports are given in Figure 5. The NIMID adopts a 'request-response' mechanism to implement the end-to-end flow control, which ensures that the data can be transferred correctly.

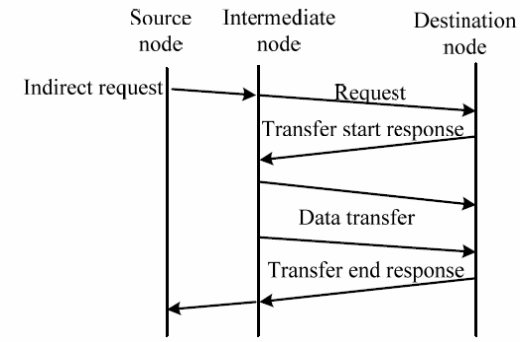
The packetisation unit is used to generate packets under the control of protocol controller unit. The packet format is illustrated in Figure 6. Three types of packets are designed to implement the transfers, which are request packet, response packet and data packet. The depacketisation unit performs the opposite function. It unpacks the data field from the packets received from the network and sends the data to RIP through the mutual interface. Moreover, if the attached on-chip network adopts adaptive routing algorithm, the packets will arrive at the destination node out of order. Thus, the NI should perform packet reordering to meet the in-order requirement of communication. The reordering

operation is designed in the NIMID as an optional extended function. In the packetisation unit, there is an order packetiser to give each packet an ordering tag. Then in the receive part, a reordering unit (Daewook et al., 2006) is set before the depacketisation unit to reorder the packets according to their ordering tag. If the network adopts deterministic routing algorithm, the logics extended for adaptive routing can be truncated to save hardware resources.

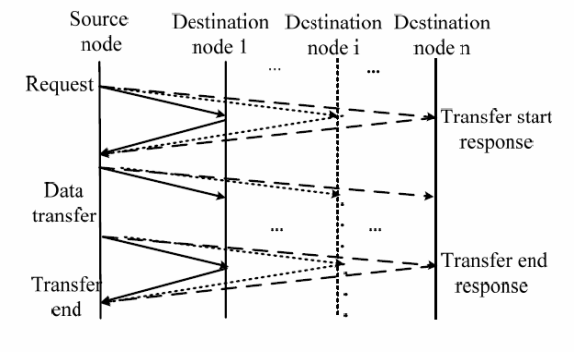
Figure 5 Transfer modes, (a) direct peer-to-peer transfer (b) indirect transfer (c) multicast transfer



(a)

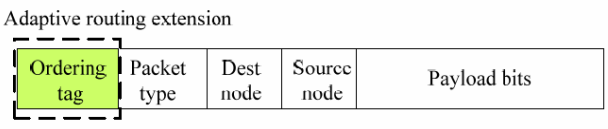


(b)



(c)

Figure 6 Format of the packet (see online version for colours)

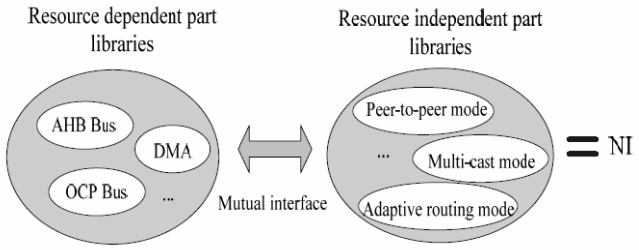


The input buffer and output buffer are used to buffer the packets and tackle the multi-clock domain synchronisation. The bit width and size of the buffers are both parameterisable to meet the requirements of various networks.

5 NI component library

In order to keep pace with the levels of integration in system-on-chip, design engineers began to embrace reuse methodology to manage the increased complexity. In the proposed NIMID design, the RDP and RIP can be decoupled owing to the standard mutual interface definitions. The design of RDP depends on interface of IP core, like data and address bus widths, control signals, etc. If a set of cores follow the same standard, then the same RDP can be used for all resources in this set. The RIP goes in the similar way. If the on-chip network possesses of the same characteristics, the RIP can be reused as well. Therefore, the authors suggest building a library consisting of components compatible with prevalent protocols (Liu et al., 2009). NoC designers can choose the optimum components from the library to construct their specified NI as illustrated in Figure 7. The design difficulty and time-to-market will be reduced as a consequence. Since the RIP is a modular structure, it can be configured at design time to meet the requirements of different types of networks. The configurable parameters include the communication mode provided by NI (e.g., peer-to-peer communication with or without multicast extension), the size of the input and output buffers, the routing technique (e.g., deterministic or adaptive), etc. The RIP library just contains the RIP units with different configurations as described in Section 4.2 and is not repeated here. This section just focuses on the construction of RDP library.

Figure 7 NI construction based on component libraries

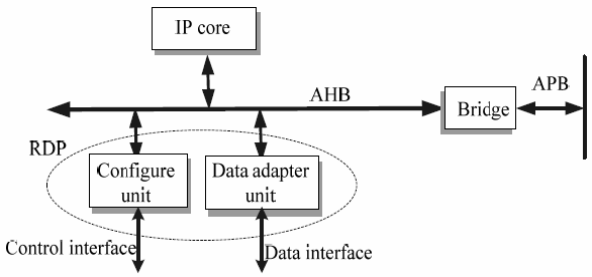


The need for effective plug-and-play design styles pushed the development of standard interface sockets (Bertozzi and Benini, 2004; Dall'Osso et al., 2003; Kistler, et al., 2006). It is common practice to design the NI that keeps backward compatibility with existing protocols, such as OCP (OCP-IP, 2003), AMBA AHB (ARM, 1999). Thus a RDP library consisting of RDP units compatible with these prevalent protocols is built. The design emphasis of RDP is imposed on how to configure the four control registers by the standard bus protocols. We take AMBA AHB and OCP attached designs as an example to illustrate the implementation of RDP.

5.1 AMBA AHB

The AMBA AHB attached RDP structure is shown in Figure 8. The configure unit is connected to AHB as a slave device. It configures the control registers introduced in Section 4.2 according to the signals received from AHB. The configure process requires three and four cycles to initiate a basic peer-to-peer and a multicast transmission, respectively. The data adapter unit is connected to AHB as a master device. It is used to coordinate the data transfer between IP core and network. The data adapter unit just transfers the signals compliant with AHB protocol to signals compliant with data interface definition.

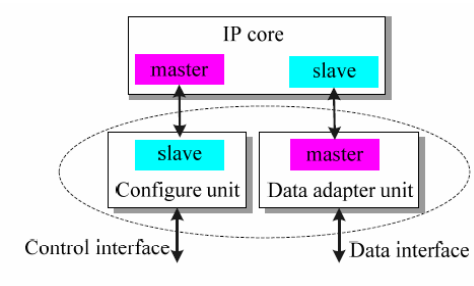
Figure 8 Structure of AMBA AHB attached RDP



5.2 OCP

The OCP-compliant NI has been implemented in Xpipes (Dall'Osso et al., 2003), which performs master and slave transmissions through two split sub-modules: NI slave and NI master. In this paper, we suggest to implement master and slave transactions in a unified structure to save hardware resources as shown in Figure 9. The configure unit is connected as a slave entity. It receives the signals from the master entity of IP core to configure the control registers in RDP. The data adapter unit is connected as a master entity, which is used to convert data format between OCP-compliant transactions and packets transferred across the network.

Figure 9 Structure of OCP attached RDP (see online version for colours)



However, if the IP core interface does not comply with any standard protocol, the RDP still can be designed by adopting the same structure. The difference is that it needs to perform configuration process and data coordination according to their own interface. The RDP structure exhibits good applicability to multiple types of IP core interface.

6 Performance evaluation

In order to evaluate the NIMID performance, this paper designed multiple types of NIs. From the perspective of resource element interface, we designed two types of RDPs: AMBA AHB type RDP and OCP type RDP. From the perspective of network, we designed four types of RIPs:

Mode 1 Supports direct peer-to-peer transfer.

Mode 2 Supports direct and indirect peer-to-peer transfers.

Mode 3 Supports mode 2 and multicast transfer.

Mode 4 Supports mode 2 and reordering mechanism.

The proposed NIMID has been implemented and synthesised with TSMC 90 nm CMOS standard cell technology under the worst case (power supply – 1.02 V, temperature – 125°C). The synthesis tool adopted is physical compiler v.2004 of Synopsys.

The delay and area performances of RDP and RIP are evaluated first. The performance of RDPs is given in Table 4, which indicates that these two types of RDPs consume almost the same timing and area. The performance of RIPs is given in Table 5. Compared to mode 1, the RIPs supporting mode 2 raises an increase of 3.3% timing overhead and 6.7% area overhead. The RIPs supporting mode 3 and mode 4 raise an increase of 8.2% on delay, 19.7% and an increase of 22.4% on area, 90.0% respectively. Since the reordering unit adopts four reorder buffer queues to implement the reorder functions (Daewook et al., 2006), much more area is consumed in this part.

Table 4 Timing and area performance of RDP

RDP	*AMBA AHB type*	*OCP type*
Delay (ns)	0.42	0.45
Area (um × um)	8,045	7,898

Table 5 Timing and area performance of RIP

RIP	*Mode1*	*Mode2*	*Mode3*	*Mode4*
Delay(ns)	0.61	0.63	0.66	0.73
Area (um × um)	31,000	33,065	37,938	58,908

Note: Buffer size: 75 bits × 8.

Then assemble these two parts to build a NI design through the mutual interface. The performance of NIMID design is provided in Table 6. The delay of different types of NIs ranges from 0.74 ns to 0.89 ns. The NI can achieve a high frequency of 1.12 GHz to 1.35 GHz.

In order to compare with other NI designs, two more types of NIMIDs are implemented and synthesised with TSMC 130 nm CMOS standard cell technology. The performance comparisons are given in Table 7. From the perspective of function, the NIMID can support more transfer modes and adaptive routing flexibly. From the perspective of performance, the NIMID can work at higher frequency. The proposed NIMID can be considered as a great competitive candidate for NoC designs.

Table 6 Performance of NIMID

Network interface		*Delay* (*ns*)	*Frequency* (*GHz*)	*Area* (*um* × *um*)	*Power consumption* (*μW*/*MHz*)	*Initiation latency* (*cycles*)
AMBA AHB type	Mode 1	0.74	1.35	45,596	13.9	3
	Mode 2	0.75	1.33	46,560	14.2	4
	Mode 3	0.78	1.28	50,106	16.0	4
	Mode 4	0.87	1.15	73,870	24.9	4
OCP type	Mode 1	0.76	1.32	44,912	12.2	3
	Mode 2	0.76	1.32	45,766	13.8	4
	Mode 3	0.78	1.28	49,289	15.0	4
	Mode 4	0.89	1.12	71,281	23.8	4

Table 7 Performance comparisons of NI

NI		*AEthereal NI*	*MANGO NI*	*NISAR NI*	*Proposed NIMID*	
Supportive function	Direct	√	√	√	√	√
	Indirect				√	√
	Multicast	√			√	√
	Reorder			√		√
Performance	Bit width (bits)	32	32	32	32	32
	Frequency (MHz)	500	483	150	650	530
	Latency (cycles)	4~6	6~8	3~4	4	4
	Area (mm^2)	0.136	0.11	0.28	0.096	0.13
	Technology (nm)	130	130	130	130	130

Note: √ – supporting; blank – not supporting.

7 Conclusions

In this paper, a novel NI method based on mutual interface definition has been proposed. It decouples the design of RDP and RIP by a set of mutual interface definitions. The NI design flexibility can be enhanced. The reusability of NI is facilitated by the method as well. Multiple NI examples compliant with AMBA AHB and OCP bus interface and capable of performing multiple transfer modes are constructed. Besides, we suggest building a set of RDP and RIP component libraries based on the proposed method. Thus designers could choose components from these two libraries to construct their NI conveniently through the mutual interface. The NI design can work at a high frequency of 1.12 GHz to 1.35 GHz with TSMC 90 nm CMOS standard cell technology. Since the work of building NI library is in the primary phase, the future work is to explore more types of RDPs and RIPs as its basic components.

Acknowledgements

This work was supported in part by the National High Technology Research and Development Program of China under Grant No. 2009AA01Z109, National Natural Science Foundation of China under Grant No. 60873112 and NSF under grant no. ECCS-0702168.

References

Ainsworth, T.W. and Pinkston T.M. (2007a) 'Characterizing the cell EIB on-chip network', *IEEE Micro*, Vol. 27, pp.6–14.

Ainsworth, T.W. and Pinkston, T.M. (2007b) 'On characterizing performance of the cell broadband engine element interconnect bus', *International Symposium on Networks-on-Chip*, pp.18–29, Princeton, NJ.

ARM (1999) *AMBA 2.0 AHB Specification*.

Atienza, D., Angiolini, F., Murali, S., Pullini, A., Benini, L. and Micheli, G.D. (2008) 'Network-on-chip design and synthesis outlook', *Journal of the VLSI Integration*, Vol. 41, No. 3, pp.340–359.

Benini, L. and Micheli, G.D. (2002) 'Network on chips: a new SoC paradigm', *IEEE Computer*, Vol. 35, No. 1, pp.70–78.

Bertozzi, D. and Benini, L. (2004) 'Xpipes: a network-on-chip architecture for gigascale systems-on-chip', *IEEE Circuits and Systems Magazine*, Vol. 4, No. 2, pp.18–31.

Bhojwani, P. and Mahapatra, R.N. (2006) 'Core network interface architecture and latency constrained on-chip communication', *IEEE Symposium on Quality Electronic Design*, pp.358–363.

Bjerregaard, T. and Mahadevan, S. et al (2005) 'An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip', *Proceedings of International Symposium on System-on-Chip*, pp.171–174.

Culler, D.J., Singh, J.P. and Gupta, A. (1999) *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, San Francisco, CA.

Daewook, K., Manho, K. and Sobelman, G.E. (2006) 'NIUGAP: low latency network interface architecture with Grey code for networks-on-chip', *Proceedings of IEEE International Symposium on Circuits and Systems*, pp.3901–3905.

Dall'Osso, M. and Biccari, G. et al. (2003) 'Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs', *International Conference on Computer Design*, pp.356–539.

Goossens, K., Dielissen, J. et al. (2005) 'Æthereal network on chip: concepts, architectures, and implementations', *IEEE Transaction on Design & Test of Computers*, Vol. 22, No. 5, pp.414–421.

Jantsch, A. and Tenhunen, H. (2003) *Networks on Chip*, pp.94–95, Kluwer Academic Publisher.

Kistler, M., Perrone, M. and Petrini, F. (2006) 'Cell multiprocessor communication network: build for speed', *IEEE Micro*, Vol. 26, No. 3, pp.10–23.

Kundu, S. and Chattopadhyay, S. (2007) 'Interfacing the cores and routers in network-on-chip using GALS', *International Symposium on Integrated Circuit*, pp.154–157.

Liu, P., Xia, B.J. et al. (2009) 'A networks-on-chip architecture design space exploration – LIB', *Computers and Electrical Engineering*, Vol. 35, No. 6, pp.817–836.

Micheli, G.D. and Benini, L. (2006) *Networks on Chips – Technology and Tools*, pp.203–284, Morgan Kaufmann Publisher.

OCP-IP (2003) *Open Core Protocol Specification*, Version 2.0, available at http://www.ocpip.org (accessed on September 2003).

Radulescr, A. and Dielissen, J. et al. (2005) 'An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 1, pp.4–17.

Xu, Y., Zhang, Q.L. et al. (2007) 'NISAR: an AXI compliant on-chip NI architecture offering transaction reordering processor', *International Conference on ASIC*, pp.890–893.