Router Design and Implementation for a Polymorphic Multimedia Processor

Ting Yang[1], Tao Li[2], Ya-Gang Wang[1], Bo-Wen Qian[2], Yu-Rong LIU[2]

Abstract. The power and design complexity of parallel processors make the inter-core communication mechanism face new requirement and challenges. This paper presents the design of the message passing mechanism and a router for a polymorphic multimedia processor, which employs two mechanisms of data communication, the shared memory among neighbouring processor elements and the remote messaging with routers. Remote messaging includes remote data transmission and remote function call. The router, which is the main remote communication mechanism, has input buffers, and employs a version of the XY routing algorithm. It is able to do efficient multicasting with fault tolerance. It adopts a specialized arbitration scheme to simplify design complexity. This improved architecture reduces the delay of inter-core communication and cuts power consumption. It also improves the performance of the polymorphic multimedia processor.

Keywords: router • polymorphic multimedia processor • fault tolerance • inter-core communication • multicast

1.1 Introduction

Parallel processing has gone through decades of research and development. Practical parallel computer architectures, from the time Stanford University put forward the thought of chip multi-processor (CMP) and the prototype of the multi-processor structure, to 2005, when Intel and AMD apply for parallel processors in large-scale, finally enter the mainstream of market now [1]. In this process, design complexity and inter-core wire delay [2] have increasingly become the core problems of designing parallel computers.

[1] Ting Yang(✉),Ya-Gang Wang
School of Computer, Xi'an University of Posts and Telecommunications, Xi’an, China
e-mail: yangting198962@163.com

[2] Tao Li, Bo-Wen Qian, Yu-Rong Liu
School of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi’an, China

In a parallel computer, each processing element executes its program fragment independently. Sometimes the program fragments need inter-core data exchange and the processing elements need to pass messages to each other. Therefore, the performance of the on-chip communication directly influences the overall performance of the parallel processor. The design optimization of routers on a network structure can greatly reduce the message delay. It is therefore the goal of this paper to explore the message passing design space to help improve the performance of our polymorphic multimedia processor [7].

At present, there are two dominant types of inter-core communication architectures [3]; one is based on bus shared cache architecture, represented by the Hydra multi-core processor [4] developed by Stanford University, and the other is based on the on-chip interconnection structure, represented by the RAW processors [5] developed by Massachusetts institute of technology.

Literature [6] proposes a low power and tolerance fault routing architecture, in which each module processes one direction(X/Y) of communication load, it needs fewer arbitration port number compared with the traditional router. Therefore, the complexity of the routing arbitration is reduced greatly.

1.2 Polymorphic Multimedia Processor Structure

The PAAG polymorphous array processor [7] employs a simple 2D mesh interconnection topology and two mechanisms for inter-core communications. The first mechanism is the near-neighbor shared memory and the second is the router based messaging network. The first mechanism is mainly used for distributed instruction level parallel computing (D-ILP) and the second is used for task parallel computation in a MIMD environment. We sometime call D-ILP operation level parallel computing (OLP) [8].

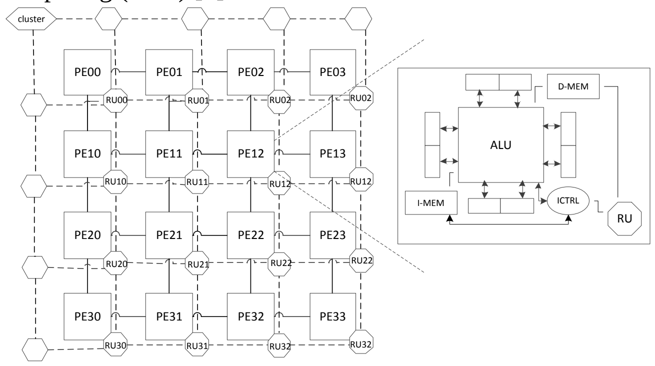

Fig. 1.1 A cluster of the PAAG polymorphic multimedia processor.

The PAAG processor consists of hierarchically organized clusters of processing elements (PEs). A 4x4 cluster is shown in Fig. 1.1. Each PE in the 4×4 cluster consists of arithmetic pipelines (ALU), instruction and data memories (I-mem and

D-mem), a router unit, a SIMD interface (ICTRL), and near-neighbour shared memories.

This paper focuses on the design and implementation of the two inter-core communication mechanisms for the PAAG multimedia processor.

1.3 Design and Implementation

1.3.1 Design and Implementation of Near Neighbour Messaging

When a PE needs data exchange with its four-neighbor PEs, it uses the shared memory. In this 2D mesh, each PE has four neighbors and the shared memory is divided into four parts as shown in Fig 1.2.

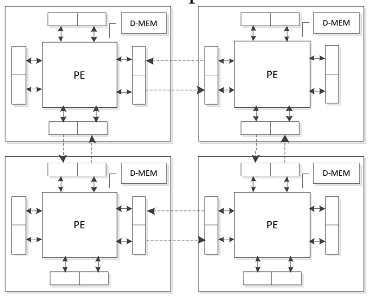

Fig. 1.2 Inter-core communication shared register

The shared memory is in the unified addressing space of a PE. An additional bit flag is attached to each 32 bit data word. The flag (valid) bit indicates data availability when operating in blocking mode.

When in blocking mode (the blocking flag is on), an instruction can proceed to the execution pipeline only if all its operands are available (i.e., the valid flags for all its operands are set). Otherwise the instruction is placed on wait until its operands become available. When the instruction finishes execution, the valid flag is reset. This mimics the data-driven data-flow [9] computation and helps implement distributed instruction level parallel (D-ILP) processing. In non-blocking mode, the valid flag has no effect.

1.3.2 Design and Implementation of Message Routers

When a PE needs exchange data with another PE that is not its immediate neighbor, or access data in cluster memory, it must use its router for remote communication. PAAG is a message passing array processor and its current version is able to address up to 1024 processing elements. The router is able to handle data exchanges and remote function calls, and it has efficient multicast capability.

A router is attached to each PE and the router works in parallel with the execution pipelines. The router may process the data from or to the local PE/remote PE/controllers of the cluster.

A PE uses special instructions MOVET (move data to remote PE), MOVEF (request data from remote PE), MVT (move data to cluster memory), MVF (request data from cluster memory) to exchange data with remote PEs. Remote function calls are served with the CALLR instruction and remote function returns with the RETR instruction. To call a (vector) SIMD function, the instruction CALLC is used and upon the finish of a SIMD function, the instruction RETC is used.

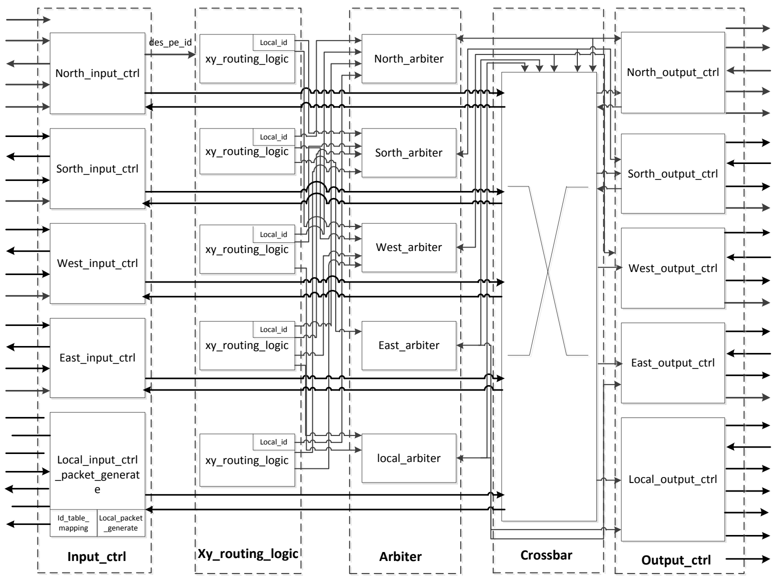

Fig. 1.3 The router hardware organization

The overall structure of the router is shown in Fig 1.3. A router is connected with four neighboring PEs and the local PE. The router hardware is divided into five parts, each of which includes an input control and packet generation module, an XY routing module, an arbiter module, and an output control module. A crossbar switch connects the five parts.

The router adopts a simple adaptive XY router algorithm to route data packets. The input control module stores incoming packet into the input FIFO. The XY routing module calculates the output port for the packet and sends the packet to the target output control module via the crossbar switch.

An adaptive routing scheme is adopted by the router. Statistics and link status are constantly monitored by the router. If a port is busy or faulty, a packet may be routed to another port instead. This provides certain degree of fault tolerance. A two stage multicast scheme is used and it is able to reach up to 12 target PEs.

Router design starts by defining the packet formats for various types of packets. The data exchange packet formats and the remote function call/return formats are given in Table 1.1 through Table 1.4.

Table 1.1 Data packet format (MOVET)

31:29	28:25	24:22	21:10	9:0
000	Size	tID	Addr	PE
31:24	23:16		15:8	7:0
B03	B02		B01	B00
		. . . . . .		
B73	B72		B71	B70

In the above, 000 is the packet type code, Size is the payload length in words, tID is the target thread ID, Addr is the target memory address and PE is the target ID. B00-B70 are payload bytes.

Table 1.2 Data request packet format (MOVEF)

31:29	28:25	24:22	21:10	9:0
001	Size	tIDr	Addr	PEr
		tIDl	Addr	PEl

Here, 001 is the packet type code, Size is the requested data length, tIDr is the remote thread ID, PEr is the remote PE ID, and Addr is the data address.

Table 1.3 Remote function call packet format (CALLR)

31:29	28:25	24:22	21:10	9:0
010		tID	Addr	PE
		rID	rAddr	rPE

Note that 010 is the packet type, PE is the remote PE ID, tID is the remote thread ID, and Addr is the program counter value of the function within the thread.

Table 1.4 Remote call return packet format (RETR)

31:29	28:25	24:22	21:10	9:0
011	Size	tID	Addr	PE
31:24	23:16		15:8	7:0
B03	B02		B01	B00
		. . . . . .		
B73	B72		B71	B70

Formats for other types of packets are neglected due to space limitation.

1.3.2.1 Design and Implementation of Input Control Module

Each of the four neighbouring directions (North\South\West\East\Local) has an input control module. The circuit of the module is shown as Fig 1.4. An incoming packet is pushed into the FIFO. The module then generates an output request symbol, and sends the destination PE ID to the XY routing module. The routing module will read the data packet and forward the packet data to the destination port via the crossbar switch.

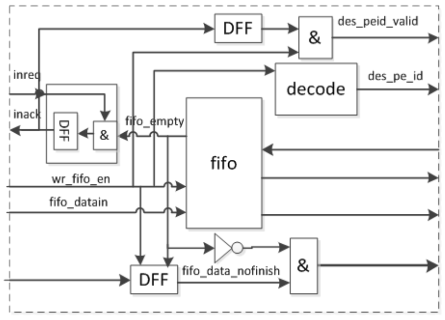

Fig. 1.4 Input Control Module Structure for north/south/west/east ports

Local input control module is just a bit more complex than the four neighbor port input control modules. Due to space limitation, the circuit for this module is not shown here.

1.3.2.2 Design and implementation of the XY routing module

According to the destination PE or controller ID, the routing module calculates the routing direction. When the destination ID is a PE, a description of the routing algorithm is given below.

1. Compare the X coordinate P_X of the PE with the packet's destination X coordinate D_X. If P_X=D_X, or if Port$_X$ is too busy or the neighbour is dead, go to step 2; otherwise, send packet along the X direction.
2. Compare the Y coordinate P_Y of the PE with the packet's destination Y coordinate D_Y. If P_Y=D_Y, send packet to local PE port; otherwise, send the packet along the Y direction.

If a packet is destined to a column (D_X) or row (D_Y) controller, the routing algorithm is as described below.

1. If the packet is for a column controller and DX is reached, send it along the Y direction; otherwise, send it along the X direction.
2. If the packet is for a row controller and DY is reached, send it along the X direction; otherwise, send it along the Y direction.

1.3.2.3 Design and implementation crossbar switch control module

The crossbar switch module selects a packet from an input port and sends it to an output port. This module will generate enable signal based on the arbitration result and the corresponding signal of output module. It will select to send FIFO read enable signal to correspond direction of input control module. And this module selects to send output data and valid signal to correspond direction of output control module based on enable signal and direction.

1.3.2.4 Design and implementation of output control module

There two types of output control modules: one type for N/S/W/E neighbour ports and one type for the local control. The router directions of output control complete the transmission of request, the reception of response, the transmission of output data and data valid signal.

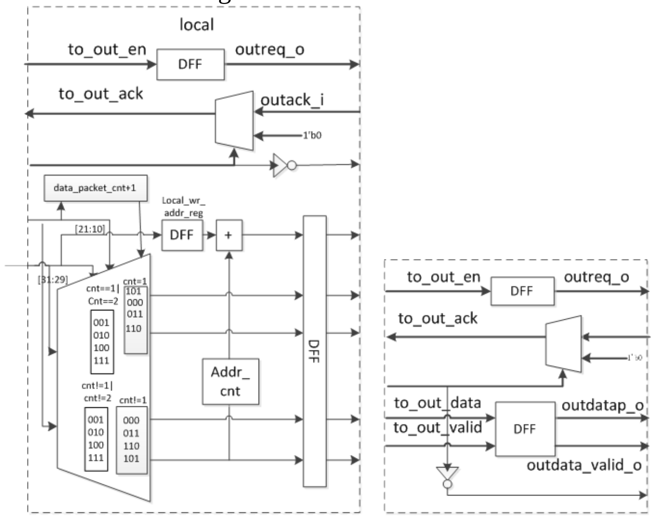

Fig. 1.8 (a) Local output control module, (b) N/S/W/E Output control module

Local output module completes the transmission of output request and the reception of corresponding output, and it decode output data format, the packet type of output. This module select control signal to send to the information local PE based on the information of packet head, including the selection of output control data and directly writing to local PE's data.

1.4 Conclusions and Future Work

This paper proposes a data communication and router architecture suitable for a polymorphic multimedia processor. The proposed routing architecture has been implemented and verified on a Xilinx V6 550 FPGA board. Two types of special data communication mechanisms have been devised, one is based on near neighbour shared memory and the other is based on a network of routers. The routing architecture can perform remote message passing and remote function call. In order to achieve low power consumption and low cost, a novel arbitration strategy using XY routing is employed. This simplifies some modules, adds fault tolerance and multicast to the architecture, and lowers energy consumption.

Future research will focus on analysing the routers throughout, average delay, maximum/minimum delay, average waiting time, channel load parameters of the router according to the traffic load of graphics and multimedia applications.

Acknowledgements I would like to thank my teacher Tao Li, who guided me and other authors who participated in this project. The support of the China Natural Science Foundation (grant 61136002) is gratefully acknowledged. The support of Shaanxi Provincial government (grant 2011k06-47) is also acknowledged.

1.5 References

1. Krste Asanovic, Ras Bodik ,et al, The Landscape of Parallel Computing Research: A View from Berkeley, University of California Berkley, Technical Report UCB/EECS-2006-183, Dec. 2006.
2. Paul Gratz, Kathikeyan, Sankaralingam, Heather Hanson. et al. Implementation and Evaluation a Dynamically Routed Processor Operand Network. Proc. IEEE 1st Int. Symp. Network-on-chip, 2007, pp.7-17
3. Xu Wei-Zhi, Song Feng-Long, Liu Zhi-Yong. etal. On Synchronization and Evaluation Method of Chipped Many-Core Professor. Chinese Journal of Computers,2010, 33(10):1777-1787 (in Chinese)
4. HAMMOND L, NAYFEH B A, OLUKOTUN K. A single-chip multiprocessor[R]. IEEE Computer, 1997, 30(9):79-85
5. KUMAR R, ZYUBAN V, TULLSEN D M. Interconnections in multi-Core architectures: Understanding Mechanisms, Overheads and Scaling. IEEE Conference Publications,2005,34(10): 408-419
6. Jongman Kim, Nicopoulos C., Dongkook Park, et al. . A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks. Proc. 33rd International Symposium on Computer Architecture，2006 , 34(2),4-15.
7. Tao Li, L. Xiao, H. Huang and J. Han, "PAAG: A Polymorphic Array Architecture for Graphics and Image Processing", Proc. 5th Int. Symp. Parallel Architectures, Algorithms and Programming (PAAP2012), Dec. 2013, Taipei, Taiwan, IEEE Computer Society CPS, pp242-249.
8. H. Huang, T. Li and J. Han, "Simulator Implementation and Performance Study of A Poly-

morphous Array Computer", 11th IEEE International Symp. on parallel Computing and applications (ISPA2013), July 2013, Melbourne, Australia, IEEE CPS, pp.1848-1855.

9. A.H.Veen, "Dataflow Machine Architecture," ACM Computing Surveys, 18(4), Dec 1986, pp.365–396.