A Low-Area Interconnect Architecture for Chip Multiprocessors

Zhiyi Yu and Bevan M. Baas
ECE Department, University of California, Davis

Abstract— **A new inter-processor communication architecture for chip multiprocessors is proposed which has a low area cost and flexible routing capability. To achieve a low area cost, the proposed statically-configurable asymmetric architecture assigns large buffer resources only to the nearest neighbor interconnect and much smaller buffer resources for long distance interconnect. To maintain flexible routing capability, each neighboring processor pair has two connecting links. Compared to a traditional dynamically-configurable interconnect architecture with symmetric buffer allocation and single-links between neighboring processor pairs, this implementation has approximately 2 times smaller communication circuitry area with a similar routing capability. Area and speed estimates are obtained with the physical design of seven chips in 0.18 μm CMOS.**

I. Introduction

Integrating multiple processors into a single chip (known as chip multiprocessors) has recently become easily achievable and common due to continuing advances in VLSI fabrication technologies. This fact makes interprocessor communication in chip multiprocessors an important design issue.

Researchers have proposed Network on Chip (NoC) solutions which use routers for inter-processor communication. Most research is based on dynamic packet-switched routing architectures [1], [2]. Another approach is the statically configurable nearest neighbor interconnect architecture used in the AsAP processor [3], where each processor communicates only with its four neighbors and long distance communication is accomplished by software in intermediate processors. IMESH [4] uses both dynamic and static interconnects. Dynamic routing architectures are flexible, but normally require relatively large circuit area and power for communication circuity. The static nearest neighbor interconnect architecture reduces area and power requirements significantly, but it results in relatively high latency for long distance communication.

We propose an *asymmetric* structure to obtain good trade offs between flexibility and cost by: treating the nearest neighbor communication and long distance communication differently, using more buffer resources for nearest neighbor connections, and using fewer buffer resources for long distance connections. Compared to traditional dynamic routing architectures, the architecture reduces communication circuitry area by approximately 2 to 4 times, while maintaining similar communication performance and flexibility.

We also found that increasing the number of links between processors is helpful to increase routing capability, but it dramatically increases processor area after a certain point which depends on implementation details. Two or three links are generally appropriate when each processor in the chip utilizes a simple single issue processor architecture.

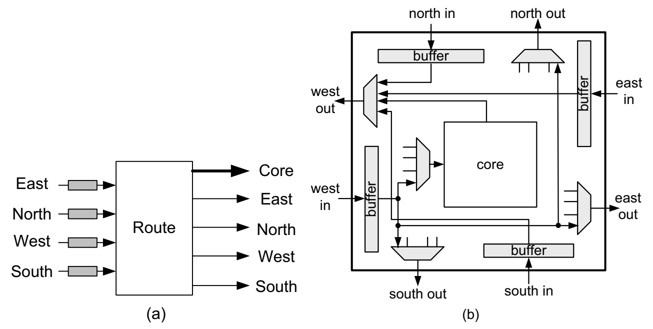

Fig. 1. (a) An illustration of interprocessor communication in a 2-D mesh, and (b) a generalized communication routing architecture in which only signals related to the west edge are drawn

II. Asymmetric interconnect architecture

A. Background: dynamic routing and static nearest neighbor interconnect

Figure 1(a) shows the interprocessor communication in a typical 2-D mesh-connected chip multiprocessor using a router architecture. The router block in each processor receives data from neighboring processors (east, north, west, and south) and then sends data to the processor core or neighboring processors. Since communication links are not always available due to slow data processing speeds or link congestion, buffers are inserted at each input edge [5]. Figure 1(b) shows a generalized diagram of the routing circuitry where only signals related to the west edge (*west in* and *west out*) are drawn. The communication logic includes four buffers and five muxes, and there is some control logic to support the communication flow control which is not shown. Other implementations are possible; for example, each output port can also have a buffer or each input buffer can be split into multiple virtual channels [1] to reduce communication latency. The area of the communication circuitry is normally dominated by the buffers and the logic in the four input/output edges is normally the same.

In many applications, communication in chip multiprocessors is localized or can be localized, which means data traffic going into the processor core is much larger than to the output paths. To minimize communication circuit overhead, another interprocessor communication strategy is to implement only the nearest neighbor interconnect logic, and the long distance

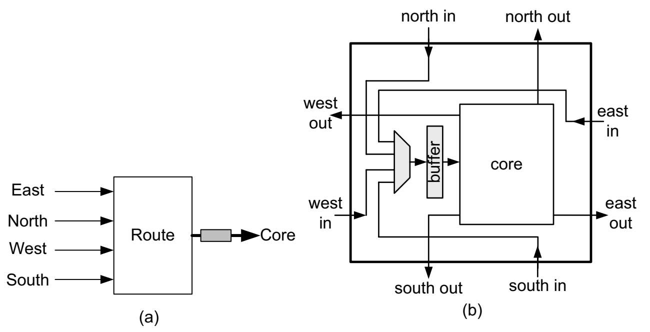

Fig. 2. The concept and circuitry diagrams of the nearest neighbor interconnect architecture. Data from four inputs are transfered only to the processing core to reduce the circuitry cost.

communication is performed by software in the intermediate processors. Figure 2(a) shows this concept of nearest neighbor interconnect and Fig. 2(b) shows its circuit diagram. Comparing Fig. 1 and Fig. 2, the nearest neighbor interconnect reduces the number of buffers from four to one and the muxes for each output port are all avoided; resulting in more than four times smaller area. But clearly it has the limitation that long distance communication places a burden on intermediate processors.

B. Asymmetrically-buffered architectures

Hu et al. have studied asymmetric data traffic patterns in inter-processor communication and proposed using different buffer resources at *input ports* of routers to match the traffic [6]. One limitation of this architecture is that for different applications as well as for different processors of individual applications, the existing asymmetric data traffic on the router's input ports is different, which makes the allocation of the buffer resources application specific. Considering the router's output ports instead of its input ports, most of the data from the input ports are delivered to the core and very few are to edges, which makes the asymmetric data traffic on the router's output more general and universal. Allocating asymmetric buffer resources at the output ports is applicable in a much wider range of applications, which is important since today, NoC architectures are used more widely than just for specific application domains.

Table I shows the data traffic of each processor for a JPEG encoder [3] which demonstrates the different asymmetric data traffic on the input and output ports of routers. On the input ports, although each processor shows a clear asymmetric communication data traffic; the major input direction for different processors are different which makes the overall traffic at the input ports quite uniform. On the output ports, however, each processor shows the similar asymmetric data traffic and overall about 80% of the data are delivered to the core. Similar results exist in other applications.

We propose an architecture which has asymmetrically-buffered output ports as shown in Fig. 3 to achieve good trade offs between cost and flexibility. As shown in Fig. 3(a), instead of equally distributing buffer resources to each output port, we allocate larger buffers to the processing core port, and smaller

TABLE I

DATA TRAFFIC OF ROUTERS IN A 9-PROCESSOR JPEG ENCODER TO PROCESS ONE 8 × 8 BLOCK ASSUMING AN ARCHITECTURE WITH FOUR-INPUTS AND FIVE-OUTPUTS AS SHOWN IN FIG. 1. DATA TO THE CORE DOMINATES TRAFFIC AT THE OUTPUT PORTS.

Proc. No.	Network data words of input ports of router				Network data words of output ports of router				
	East	North	West	South	Core	East	North	West	South
1	0	64	0	0	64	0	0	0	0
2	0	64	0	0	64	0	0	0	0
3	0	64	0	0	64	0	0	0	0
4	0	0	64	0	64	64	0	0	0
5	0	0	96	0	64	0	32	0	0
6	0	0	0	64	1	0	0	63	0
7	0	0	0	3	3	0	0	0	0
8	63	0	0	0	63	0	0	0	0
9	4	0	0	252	256	0	0	0	0
Total	67	192	160	319	643	64	32	63	0
Relative	9%	26%	22%	43%	**80%**	8%	4%	8%	0%

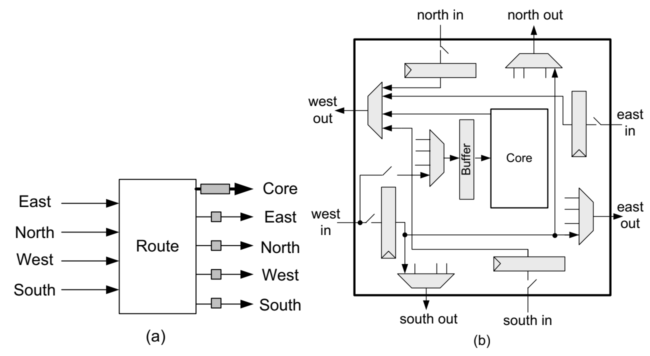

Fig. 3. The concept and circuitry diagrams of the proposed inter-processor communication architecture; it has the asymmetric buffer resource for the long distance interconnect and the local core interconnect

buffers (one or several registers) to the other ports. Figure 3(b) shows the circuitry diagram where only signals related to the west edge (*west in* and *west out*) are drawn. This scheme's circuit area is similar to the nearest neighbor interconnect by adding a few registers and muxes. From the point of view of routing capability, this scheme is similar to the dynamic routing architecture, since reducing the buffers in ports for long distance communication does not significantly affect system performance when the communication is localized. When using one large buffer for the processing core, the proposed architecture can save about 4 times the area compared to the Fig. 1 architecture. If using two large buffers, the area savings is about 2 times lower.

III. SINGLE LINK VS. MULTIPLE LINKS

Using the architecture shown in Fig. 3, there are still many options for the communication logic realization. For analyses presented in this paper, we assign two ports (buffers) for each processing core, and use a static routing architecture due to its low cost.

One of the key differences between on-chip and inter-chip interconnects is that there are more wire resources on chip, while inter-chip connections are normally limited by the

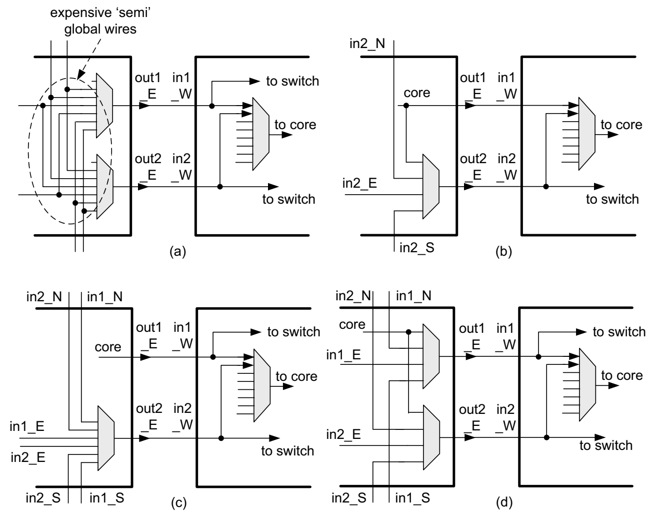

Fig. 4. Four inter-processor interconnect schemes utilizing double links: (a) fully connected; (b) separated nearest neighbor link and long distance link; (c) separated link from core and link from edges; and (d) connections exist between 'corresponding' links. Methods (b), (c), and (d) correspond to options 12, 15 and 44 in Table II.

TABLE II

INTERCONNECT ARCHITECTURE OPTIONS FOR DOUBLE LINKS. *Yes* MEANS A CONNECTION EXISTS BETWEEN INPUT AND OUTPUT, *No* MEANS NO CONNECTION EXISTS, AND *xx* MEANS DON'T CARE.

options	in1-out1	in2-out1	core-out1	in1-out2	in2-out2	core-out2
1-8	No	No	No	xx	xx	xx
9	No	No	Yes	No	No	No
10	No	No	Yes	No	No	Yes
11	No	No	Yes	No	Yes	No
12	No	No	Yes	No	Yes	Yes
13	No	No	Yes	Yes	No	No
14	No	No	Yes	Yes	No	Yes
15	No	No	Yes	Yes	Yes	No
16	No	No	Yes	Yes	Yes	Yes
....						
43	Yes	No	Yes	No	Yes	No
44	Yes	No	Yes	No	Yes	Yes
45-48	Yes	No	Yes	Yes	xx	xx
49-64	Yes	Yes	xx	xx	xx	xx

available chip IO pins. Dally and Towles [1] suggest increasing the wordwidth of each link in NoCs to take advantage of this fact. We explore another option in this section of increasing the number of links at each edge to increase connection capability and flexibility.

A. Increasing the number of links

The overhead of increasing the number of links is high not only because of necessary control logic, but more importantly because of semi-global wires inside each processor which affect system area/speed/power significantly. Fig. 4(a) shows the fully-connected two-link architecture.

Methods are available to simplify the fully connected architecture. Considering the router's logic at the east edge which receives data from *North*, *West*, *South* and *Core* and sends to *East* output, we use *in1_N, in2_N, in1_W, in2_W, in1_S, in2_S, sig_core, out1_E,* and *out2_E* to define these signals. A large exploration space exists at first glance since 7 inputs and 2 outputs have 2^{14} connection options. Since three input edges are symmetric, we group $\{$in1_N, in1_W, in1_S$\}$ together as input in1 and another input group named as in2, so the input number is reduced to 3 and the exploration space is reduced to $2^6 = 64$ as shown in Table II. In options 1-8, out1 does not have any connections so they are not considered as two-links architectures. Option 9 is neglected for a similar reasons. Option 10 only connects the processing core to the outputs and it is essentially the same as the nearest neighbor architecture. By examining all other options, we found that options 12, 15 and 44 are potentially good choices, and their corresponding circuit diagrams are shown in Fig. 4(b), (c) and (d) respectively.

In terms of the area cost and the routing flexibility of the four architectures shown in Fig. 4, architecture (a) has the most flexible connection while it has the biggest cost; architecture (b) has the most strict connection limit while it has the smallest cost; architecture (c) has the connection flexibility and cost in between (a) and (b). Architecture (d) has the area cost similar with (c), and interestingly, its routing capability is the same with architecture (a) by setting routing paths. We investigate further the Fig. 4(d) architecture due to its routing flexibility and moderate circuitry cost, and Fig. 4(b) due to its small circuitry cost.

A more quantitative analysis is given in the following subsections. The architectures we evaluate include the single-link architecture, the double-links architectures shown in Fig. 4(b) and (d), and three and four links architectures which are enhanced versions of Fig. 4(b) and (d).

B. Area and speed estimates from seven chip designs

Increasing the number of communication links requires additional control logic, which increases circuitry area and affects processor speed. We have designed seven processor chips with each containing different communication circuits in a 0.18 μm technology, synthesizing from RTL code and completing the physical layout using Cadence Encounter. Standard cell area utilization is initially 70% and typically increases to about 85% after clock tree insertion and inplace optimization. The seven types are as follows: (1) single-link, (2) double-link in Fig. 4(b), (3) double-link in Fig. 4(d), (4) three-link version of Fig. 4(b), (5) three-link version of Fig. 4(d), (6) four-link version of Fig. 4(b), and (7) four-link version of Fig. 4(d).

Fig. 5 shows key data for the seven chip designs. Types 6 and 7 (four link architectures) have a noticeable nearly 25% increase in area because they can not successfully complete routing until the area utilization is reduced to 64% and 65% respectively. This result provides interesting insight into how many global wires can fit into a chip. For a processor with a 0.66 mm^2 area and a 0.8 mm edge, assuming a minimum 1 μm pitch between IO pins, an optimistic estimation is that each edge can fit 800 IO pins consuming one perpendicular metal layer; beyond this range the processor size will become

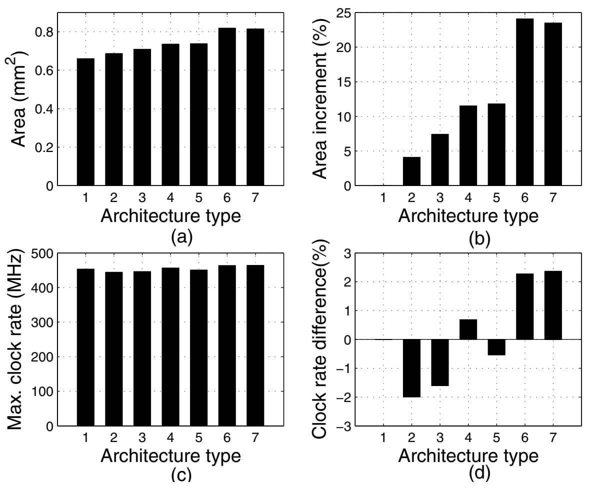

Fig. 5. Comparison of designed processors with the seven communication link types showing: (a) absolute area; (b) area relative to type 1; (c) absolute speed; and (d) speed relative to type 1

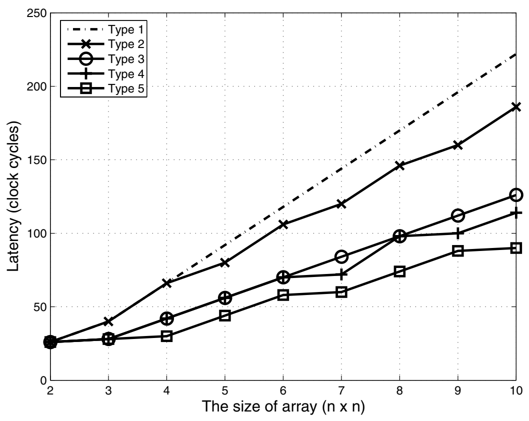

Fig. 6. Comparing the communication latency (clock cycles) of interconnect architectures type 1 to 5, by varying the size of the array and using all-to-one communication. Type 6 and 7 architectures are not included in the comparison due to their high area cost as shown in Fig. 5

IO pin or wire dominated. This estimation can be true if these IO pins are all connected to short wires. For global wires used for communication routers, increasing the number of wires will quickly result in routing congestion and increase the processor size. In our example, each processor edge in four-link architectures has about 160 IO pins, much less than the optimistic 800 IO pins. Four or more communication link architectures are less desirable due to their high area cost.

Fig. 5 (c) shows the processor's speed and Fig. 5 (d) shows the relative speed difference of each architecture compared to the type 1 architecture. Each processor has similar speed and the difference is within 2%.

C. Performance

We use basic communication patterns including one-to-one, one-to-all, all-to-one and all-to-all to evaluate the performance of architectures with different numbers of links.

For one-to-one or one-to-all communication, each processor only requires one source so that the single link architecture is sufficient and has the same communication latency with other architectures. A little surprisingly, all architectures have the same latency for all-to-all communication, because each processor needs data from both horizontal and vertical neighbor processors and both of the two buffers (ports) of each processor are occupied, prohibiting the usage of the additional links for long distance communication.

These architectures have different features in all-to-one communication as shown in Fig. 6. For the type 1 (single link) architecture, most or all of the link resources are occupied by the nearest neighbor interconnect and little can be used for the direct routing, so the latency is relatively high. Increasing the number of links helps when the latency is limited by the link resources. Type 2 (double links with separated nearest neighbor link) has little advantage to type 1 with a relatively much higher area, and type 4 (three links with separated nearest neighbor link) has little advantage to type 3 (double links) but with a relatively much higher area, so that type 2 and 4 are not considered. Type 3 architecture is about two times faster than the single link architecture, which makes it a good candidate. Comparing type 5 (three links) architecture with type 3, they have the same latency within a small communication domain (2 $\times$ 2 and 3 $\times$ 3 arrays), while the three-link architecture benefits when the array grows.

IV. Conclusion and Acknowledgments

An asymmetric inter-processor communication architecture which assigns more buffer resources to the nearest neighbor interconnect and fewer buffer resources to the long distance interconnect is proposed. Using two or three links at each edge achieves good area/performance tradeoffs for chip multiprocessors containing simple single issue processors; and the optimal number of links is expected to increase if larger processors are used.

The authors thank members of the VCL, R. Krishnamurthy, M. Anders, and S. Mathew; and gratefully acknowledge support from Intel, UC MICRO, NSF Grant No. 0430090 and CAREER award 0546907, SRC GRC Grant 1598, ST Microelectronics, IntellaSys, S Machines, and Artisan.

References

[1] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *IEEE International Conference on Design Automation*, June 2001, pp. 684–689.

[2] S. Vangal, J. Howard, G. Rule, et al., "An 80-tile 1.28TFLPS network-on-chip in 65nm CMOS," in *ISSCC*, Feb. 2007, pp. 98–99.

[3] Z. Yu, M. Meeuwsen, R. Apperson, et al., "An asynchronous array of simple processors for DSP applications," in *ISSCC*, Feb. 2006, pp. 428–429.

[4] D. Wentzlaff, P. Griffin, H. Hoffmann, et al., "On-chip nterconnection architecture of the tile processor," *IEEE Micro*, pp. 15–31, Sept/Oct 2007.

[5] S. Vangal, A. Singh, J. Howard, et al., "A 5.1GHz 0.34mm^2 router for network-on-chip applications," in *Symposium on VLSI Circuits*, June 2007, pp. 42–43.

[6] J. Hu and R. Marculescu, "Application-specific buffer space allocation for network-on-chip route design," in *ICCAD*, 2004, pp. 354–361.