Building Asynchronous Routers with Independent Sub-Channels

Wei Song and Doug Edwards
School of Computer Science, University of Manchester, Manchester M13 9PL UK
{songw, doug}@cs.man.ac.uk

***Abstract*—Network-on-chip (NoC) has been used as the new on-chip communication paradigm. Asynchronous NoCs are power efficient and robust to process variation but they are slow. One reason for the low speed is the way that asynchronous routers use to build wide channels. To meet the bandwidth requirement, current routers broaden their channels by synchronizing multiple sub-channels. The C-element and buffer trees introduced by the synchronization increase the cycle period. A new router is proposed to use multiple independent sub-channels to transmit data. Since the synchronization is removed, the cycle period of all sub-channels are reduced speeding up the network. Two routers, one using multiple independent sub-channels and one using the synchronized wide channel, are implemented at the layout level. The simulation results show that the new router using multiple independent sub-channels reduces the router latency and the cycle period.**

I. Introduction

Network-on-chip (NoC) [1] provides a scalable on-chip communication architecture for current multi-processor system-on-chip (MPSoC) systems. An on-chip network can be classified as a synchronous network with data synchronized by a global clock, a mesochronous network with data transmitted across clock domains triggered by phase-shifted clocks, or as an asynchronous network with data delivered by handshakes. Because synchronous and mesochronous networks use a predefined clock to synchronize data and the EDA tools are mature, they are fast and area efficient. Compared with them, asynchronous networks are slow and large. The handshake circuits increase the area overhead and slow down the data transmission. The lack of EDA support also hinders their adoption by industry.

Asynchronous networks, however, have their own advantages. They are power efficient because they remove the power consuming clock tree. Delay insensitive asynchronous circuits are naturally tolerant to process variation, which is a major problem for nanoscale synchronous circuits. The asynchronous network divides the whole chip into several isolated clock domains, which unifies the network interface and shortens the overall design time.

Although asynchronous networks tend to be slow, their special benefits are crucial to nanoscale MPSoC systems. The speed of these networks could be improved by modifying the current router structures. To meet the bandwidth requirement, current asynchronous NoCs usually broaden the data width of their quasi delay insensitive (QDI) asynchronous channels by synchronizing more sub-channels [2], [3], [4], in a similar manner to increasing the wire count of a bus in synchronous circuits. However, this synchronization introduces a speed penalty to asynchronous channels as explained below.

Synchronous circuits work under the timing assumption that data arrive earlier than the synchronization signal, the global clock. Increasing the data width means more flip-flops on a pipeline stage. Because the clock tree is excluded from the critical paths, no significant delay is introduced; although the clock skew problem is worsened, this is usually manageable. However, QDI channels are driven by handshake circuits and the completion detection circuits are included in the critical paths. Increasing the data width means more sub-channels are synchronized by a larger completion detection circuit incurring extra latency.

The speed penalty can be alleviated if a channel is divided into multiple independent sub-channels because the completion detection circuit is divided into several smaller ones. In this paper, we propose a new router to utilize the independent sub-channels. To evaluate the effect of using independent sub-channels, two routers, one using multiple independent sub-channels and one using synchronized channels, are implemented and tested. The simulation shows that the channel division speeds up the router.

The remainder of this paper is organized as follows: section II explains the speed penalty of the channel synchronization, section III proposes a new wormhole router using multiple independent sub-channels, section IV compares the implementations of routers with and without channel division, and the paper is concluded in section V.

II. Channel Period with its Data-Width

A. Asynchronous circuits styles

Many handshake protocols could be used to build asynchronous circuits but only some of them are suitable for asynchronous router designs.

The 4-phase bundled-data protocol has been used in MANGO [5] and QNoC [6]. Since the bundled-data protocol is not delay insensitive, circuits need cautious delay constraints and extra delay lines [7]. The bundled-data protocol is not preferred for channels between routers.

The 4-phase 1-of-n protocol family has been used in CHAIN [3] and ANoC [2]. The 1-of-n protocol family is QDI, therefore, 1-of-n protocols are tolerant to process variation. In practice however, because the number of wires grows

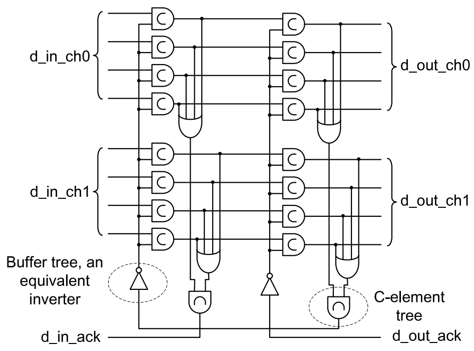


Fig. 1. A 4-bit 1-of-4 asynchronous channel

exponentially with the data path, 1-of-4 is the largest 1-hot code normally used.

To increase the data width with less area overhead, the QDI m-of-n protocol fires m bits of a total n wires on one pipeline stage each cycle [4], which has been used in SpiNNaker [8]. However, extra encoders and decoders are needed to translate binary data to and from m-of-n codes. NoC is a distributed arbitration environment where flits are analyzed in each router along the path. As a result, one decoder is necessary on each port of each router giving a large area overhead. For SpiNNaker NoCs, the 2-phase 2-of-7 protocol is used only on channels between chips and the router on-chip is a synchronous module processing the data after the synchronous to asynchronous interface.

Because the bundled-data protocol needs delay estimation and m-of-n protocols introduce extra decoders and encoders, the QDI 1-of-n protocol is preferred for asynchronous router designs.

B. The delay penalty

Fig. 1 shows a 4 bit asynchronous channel formed by synchronizing two 1-of-4 channels. The synchronization slows down the channel for three reasons: firstly, a C-element tree is added on the receiver side to synchronize all sub-channels and generate the ack signal. This increases the latency of the backward path and the channel period. Secondly, because the fan-out of the ack signal is increased, a buffer tree is automatically added by synthesis tools on the sender side to drive all C-elements equally. This buffer tree also increases the latency of the backwards path. Thirdly, the synchronization forces all sub-channels to wait for the last arrived data, which is the worst latency. C-elements are recognized as combinational circuits by synthesis tools, therefore, the skew of the ack signal to each C-element is not optimized and the buffer tree added will not be balanced. An unbalanced buffer tree incurs different input transition time for each C-element and because C-elements are not recognized as storage elements by layout tools, it is difficult to set timing constraints to data paths. The length of the wires between two pipeline stages are unlikely to be identical. Because of the aforementioned points, asynchronous pipelines are likely to suffer worse delay differences on data paths than synchronous pipelines.

For the channels inside a router, since the channel period is dominated by cell delay, the C-element tree and the buffer tree are the major causes of the period enlargement. It is worse for the channels between routers. The difference between latencies among long interconnects is added to the extra latency introduced by cell delay that reduces the speed.

This speed penalty is a common outcome of increasing the data width of a channel in all asynchronous circuit styles. Because 1-of-n codes are in practice limited to 1-of-4, the maximum channel width is 2 bits. Channel with a data width larger than 2 bits is built by synchronizing multiple single channels introducing a speed penalty. Although the m-of-n protocol transmits more data than the 1-of-n protocol, the completion detection circuit is substituted by a more complex tree structure [4] than the simple OR gate in Fig. 1, which has the same latency effect as the C-element tree. Besides, constrained by the area overhead of the detection circuit, the encoder and the decoder, the maximal data width of a single m-of-n channel reported is 4 bits in the 2-of-7 case of SpiNNaker [8]. Any m-of-n channel that has a data width larger than 4 bits still requires a C-tree to synchronize multiple single channels.

The bundled-data protocol has a smaller speed penalty than delay insensitive protocols, since the completion detection circuit is removed through timing constraints. However, the buffer tree added to trigger C-elements (latches) is retained. An asynchronous router using the bundled-data channels has reported a similar penalty of 0.2% per bit performance degradation [6].

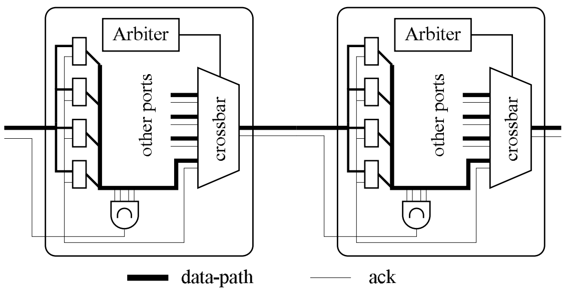
III. A Wormhole Router Using Independent Sub-Channels

In the previous section, we have demonstrated the speed penalty of simply synchronizing the sub-channels. However, the idea of using independent sub-channels also brings control problems. Data transmitted on sub-channels are still required to be synchronized. They may not be synchronized every cycle but at least at the start of a frame.

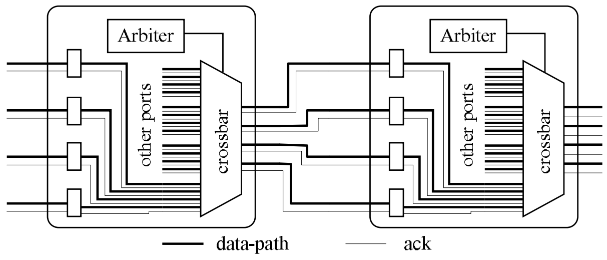
A. Reduce the synchronization

The data path of a wormhole NoC using synchronized channels could be simplified into Fig. 2(a). Assuming the asynchronous channel between routers is formed by four sub-channels, a four input C-element tree is required to generate the ack signal on each port. All sub-channels are merged into one channel and traverse the router through the multiplexer controlled by an arbiter. To remove the C-element tree, the data path could be restructured as shown in Fig. 2(b). The four sub-channels still go through the multiplexer together but each of them has its own ack line. They could run independently.

However, sub-channels must still be synchronized. Recalling the flow control method in the wormhole network, a frame comprises a head flit, several data flits and a tail flit. The head flit is sent first. Routers in the network analyze the address information in the head flit and reserve a path by configuring the multiplexer in their crossbars. Data flits then follow the

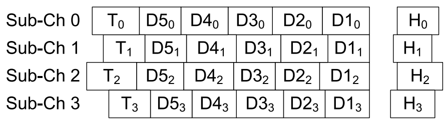


(a) The data path of the traditional wormhole NoC



(b) The data path using multiple sub-channels

Fig. 2. Remove the synchronization



Sub-Ch 0 T_0 $D5_0$ $D4_0$ $D3_0$ $D2_0$ $D1_0$ H_0
Sub-Ch 1 T_1 $D5_1$ $D4_1$ $D3_1$ $D2_1$ $D1_1$ H_1
Sub-Ch 2 T_2 $D5_2$ $D4_2$ $D3_2$ $D2_2$ $D1_2$ H_2
Sub-Ch 3 T_3 $D5_3$ $D4_3$ $D3_3$ $D2_3$ $D1_3$ H_3

Fig. 3. A modified flow control method using four sub-channels

path reserved by the head flit. Finally the path is released by the tail flit after the data.

Adhering to the basic flit sequence, Fig. 3 shows the modified wormhole flow control method using the independent sub-channels. Suppose a channel is divided into four sub-channels indexed from 0 to 3. The data of a flit are allocated on sub-channels in bit-order. The head flit is still sent first. The address in the head flit is received and analyzed by sub-channel 0. Before the address is analyzed and the crossbar is configured, all data flits are blocked. Once the configuration is done, all data flits are transmitted through the sub-channels independently. This modified flow control method reduces the synchronization among sub-channels from once per cycle to once per frame, which speeds up the data transmission.

Without the synchronization among sub-channels, the data on different sub-channels could be out of order. For example, if sub-channel 0 in Fig. 3 is much slower than channel 1, data block $D3_1$ may reach the target node earlier than data block $D2_0$. However, this could be solved by the network interface in the target node. Since the size of a flit is fixed and the data blocks on each sub-channel are in order, the network interface could easily calculate the flit index and store the data by a memory interface or it could use four FIFOs to receive the flits and reunite the frame later. Note that the four FIFOs structure introduces no storage overhead but extra control logic. The total size of FIFOs is kept the same.

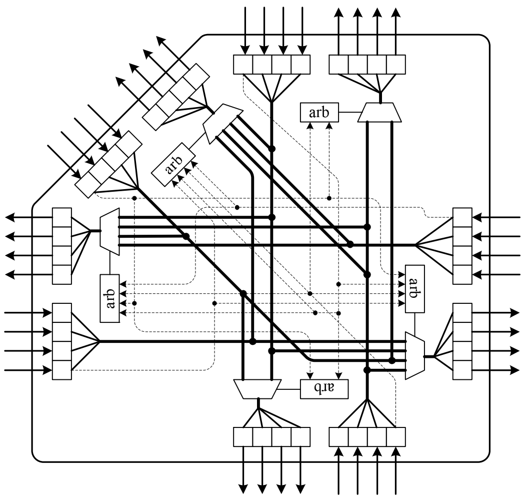


Fig. 4. The new router with four sub-channels on a port

Although sub-channels start together, they end separately. The multiplexer is unlikely to be released partially. Some extra control logic is added to guarantee that no further flit is transmitted on a sub-channel after the tail flit and the multiplexer is released when the tail flit is received by all sub-channels.

The modified flow control method reduces the synchronization among sub-channels to once per frame, which allows sub-channels to run independently during the data session and reduces the total transmission time.

B. The router architecture

We have implemented a traditional wormhole router using synchronized channels and a new router using independent sub-channels. Both routers have five 32 bit ports. For the traditional router, the wire count of the data channel is 67 wires including 64 data wires, two wires for the end-of-frame bit and an ack wire. To avoid separating the address in the head flit, the data width of a sub-channel in the new router is set to 8 bits, therefore, the address received by sub-channel 0 is enough to address a 16x16 mesh. Consequently, the 32 bit channel is divided into four 8 bit sub-channels and the wire count is increased to 76 because the sub-channels now have their own set of end-of-frame and ack wires.

The architecture of the new router is shown in Fig. 4. The router comprises five input buffers, five output buffers and five multiplexers controlled by five arbiters. The depth of all buffers are one bit. Since a channel has four sub-channels, all input buffers and output buffers are divided into four parts with their own end-of-frame detection circuits. Routing decisions are made by sub-channel 0, therefore, only the buffer of sub-channel 0 is connected to arbiters.

According to the modified flow control method, sub-channel 0 analyzes the address in the head flit and makes a request on one of the connected arbiters. After a path is granted, sub-channel 0 informs all the other sub-channels and the data session starts. When a sub-channel has detected the tail flit

TABLE I
THE AREA OF ROUTERS IN GATE COUNT

	Traditional	New	Overhead
Input Buffers	16.42K	10.02K	-39.0%
Output Buffers	3.04K	3.42K	12.5%
Arbiters	0.45K	0.45K	0%
Multiplexers	1.51K	1.67K	10.6%
Total	21.41K	15.56K	-27.3%

unit: area of a NAND2_X1 standard cell

through a '1' on the end-of-frame bit, it stops sending new flits, informs the situation to channel 0 and waits for the grant of a new path.

The traditional router has the similar structure but the buffers and channels are not divided.

IV. Performance Comparison

A. Hardware implementation

The major parts of the routers are designed in Balsa [9], an asynchronous circuit language and synthesizer. Multiplexers and output buffers are manually implemented to reduce the area overhead. Both the manually written netlist and the netlist generated by Balsa are in Verilog HDL using a 45 nm open source cell library [10]. Both routers are synthesized and implemented into layouts using standard EDA tools.

Table I shows an interesting area comparison result. Dividing a channel into four sub-channels should increase the total area because sub-channels have their own set of end-of-frame and ack wires along with the end-of-frame detection circuits. This is demonstrated by the area of the multiplexers and the output buffers. However, the input buffers of the new router is 39.0% smaller than the traditional one, which leads to a 27.3% reduction to the total area.

The circuit generated by Balsa is control driven [11]. Data paths are synchronized by control circuits, therefore, the size of glue logic between data path and control logic, such as the MUX, DEMUX and read control of variables, are linear with the wire count of data paths. The channel division reduces the wire count of data paths in each sub-channel. Because the routing decision is made by sub-channel 0 in the new router, the area of glue logic are reduced to 1/4. As a result, the reduction of glue logic in input buffers compensates the area overhead introduced by the channel division.

B. Latency estimation

Table II shows the latency of the routers after layout. The time results are obtained from measuring a frame travelling from the south input port to the east output port. Derived from the XY routing algorithm, the south input buffer can send data to four output buffers and the the east output buffer may receive data from four input buffers. Both of them are the worst connection scenario leading to the worst latency.

Because routing decisions are made by the head flits, they have a longer cycle period than data flits. Due to the reduced C-element and buffer tree, the new router demonstrates a 1.0 ns shorter head flit period than the traditional one. Althrough the route decision is made by sub-channel 0, sub-channels of the new router show no significant speed difference. The period of sub-channels for data flits is around 4.1 ns, 0.9ns smaller than the synchronized channels. For the router latency of data flits, the new router also shows 0.5 ns reduction.

TABLE II
LATENCY OF ROUTERS

	Traditional	New	Saving
Period for the head flit	6.5 ns	5.5 ns	1.0 ns
Period for data flits	4.9 ns	4.1 ns	0.8 ns
Latency for data flits	1.1 ns	0.6 ns	0.5 ns

V. Conclusion

In this paper, we have analyzed the speed penalty of implementing an asynchronous channel by synchronizing multiple sub-channels. We have proposed a new router that allows sub-channels to transmit data independently after the head flit.

Two routers, one using 4 independent 8 bit sub-channels and one using a 32 bit channel, have been implemented. The implementations show that using sub-channels reduces the cycle period of data flits and the head flits by 0.8 ns and 0.5 ns respectively.

Acknowledgements

This work is supported by EPSRC grant EP/E06065X/1. The authors would like to thank Luis Tarazona at the University of Manchester for his help on the speed optimization of Balsa and the advice from all the anonymous referees.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. of DAC*, 2001.

[2] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework," in *Proc. of ASYNC*, March 2005, pp. 54–63.

[3] J. Bainbridge and S. Furber, "Chain: a delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, pp. 16–23, 2002.

[4] J. Bainbridge, W. Toms, D. Edwards, and S. Furber, "Delay-insensitive, point-to-point interconnect using m-of-n codes," in *Proc. of ASYNC*, May 2003, pp. 132–140.

[5] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. of DATE*, 2005, pp. 1226–1231.

[6] R. Dobkin, R. Ginosar, and A. Kolodny, "QNoC asynchronous router," *Integration, the VLSI Journal*, vol. 42, no. 2, pp. 103–115, 2009.

[7] M. Singh and S. M. Nowick, "MOUSETRAP: ultra-high-speed transition-signaling asynchronous pipelines," in *Proc. of ICCD*, 2001, pp. 9–17.

[8] L. A. Plana, S. B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, and S. Yang, "A globally asynchronous, locally synchronous infrastructure for a massively-parallel multiprocessor," *IEEE Design and Test of Computers*, vol. 24, no. 5, pp. 454–463, 2007.

[9] D. Edwards and A. Bardsley, "Balsa: an asynchronous hardware synthesis language," *The Computer Journal*, vol. 45, no. 1, pp. 12–18, 2002.

[10] "the Nangate 45nm Open Cell Library." [Online]. Available: http://www.nangate.com/

[11] S. Taylor, D. Edwards, and L. Plana, "Automatic compilation of data-driven circuits," in *Proc. of ASYNC*, April 2008, pp. 3–14.