A Low Latency Wormhole Router for Asynchronous On-chip Networks

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

Abstract—Asynchronous on-chip networks are power efficient and tolerant to process variation but they are slower than synchronous on-chip networks. A low latency asynchronous wormhole router is proposed using sliced sub-channels and the lookahead pipeline. Channel slicing removes the C-element tree in the completion detection circuit and converts a channel into multiple independent sub-channels reducing the cycle period. The lookahead pipeline uses the early evaluation protocol to reduce cycle period. Using the lookahead pipeline on the pipeline stages with the maximal cycle period improves the overall throughput. The router is a pure standard cell design implemented by a 0.13 μm technology. The cycle period of the router at the typical corner is 1.7 ns, providing 2.35GByte/sec throughput per port.

I. Introduction

Network-on-chip [1] is the state-of-the-art on-chip communication fabric for current multi-processor SoC systems. The on-chip network could be a synchronous network where routers are driven by a global clock, or an asynchronous network where routers are self-timed circuits connected by asynchronous pipelines. Thanks to mature EDA tools and the timing assumptions allowed by the global clock, synchronous networks are fast and area efficient but the clock tree is power consuming [2]. By contrast, the clock-less asynchronous networks are comparatively slow but power efficient. In addition, they are tolerant to process variation and could divide 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 advantages are crucial to nanoscale SoC systems. In this paper, a low latency asynchronous router is designed using two novel techniques: channel slicing and the lookahead pipeline [3].

Channel slicing: The state-of-the-art quasi delay-insensitive (QDI) pipelines in routers are built by synchronizing multiple bit-level pipelines (sub-channels) [4, 5, 6, 7, 8]. The C-element tree in the completion detection circuit (synchronization circuit) increases the cycle period and reduces throughput. Instead of synchronizing sub-channels, we propose to use sub-channels in parallel. Since the C-element tree is removed, sub-channels run faster than the synchronized channel. Extra controllers are added to resynchronize sub-channels during special intervals, such as the route decision procedure.

Lookahead pipeline: The lookahead pipeline is an improved dual-rail pipeline using an early evaluation protocol, proposed by Montek [3]. It is not QDI but the timing assumptions are satisfiable and it could be used to reduce the period of the critical cycle (pipeline stages with the maximal cycle period).

In this paper, the router is implemented using a 0.13 μm standard cell library and the cycle period is 1.7 ns, providing 2.35 GByte/sec throughput per port.

The remainder of this paper is organized as follows: section II describes the general architecture of the on-chip network. Section III explains how channel slicing and the lookahead pipeline can improve speed. Section IV demonstrates the detailed implementation of the router. Section V shows the simulation results of the implementation, analyzes the effect of the two techniques used, illustrates the impact of pipeline data width on area and speed, and compares our router with other published asynchronous router designs. Finally the paper is concluded in section VI.

II. Network Architecture

Fig. 1(a) shows a globally asynchronous and locally synchronous (GALS) [2] network. A network node (Fig. 1(b)) comprises a processing element (PE), a network interface (NI) and a router (RT). The processing element could be a local system controlled by a processor or a hardware IP running a specific function. Serving as a slave device to the processing element, the network interface provides a duplex channel for the processing element to communicate with the chip level asynchronous network. To ease the network communication, the network interface splits the frames generated by the local processing element into a sequence of flits of fixed length before sending them to routers. It also regroups the received flits into frames before delivering them to the local processing element. In a GALS network, the network interface also serves as a synchronous/asynchronous adaptor to ensure the faultless cross timing domain data transmission [9, 10, 11]. Similar to the routers used in macro networks, routers in on-chip networks are distributed route deciders and message delivers but with tighter area budget and higher throughput requirement. They are fully asynchronous circuits in the proposed GALS network.

This paper concentrates on the wormhole flow control method and the hardware implementation of asynchronous routers; therefore, all other design aspects are set to broadly accepted configurations. A mesh topology is used due to its easy mapping on a 2-D layout. Frames are routed by the XY dimension order routing algorithm. Nodes in the network are identified by a (x, y) address shown in Fig. 1(a). Network interfaces have enough buffer space to guarantee that a flit is consumed by a network interface in finite time. The network is assumed to be error-free so that no deadlock or livelock occurs.

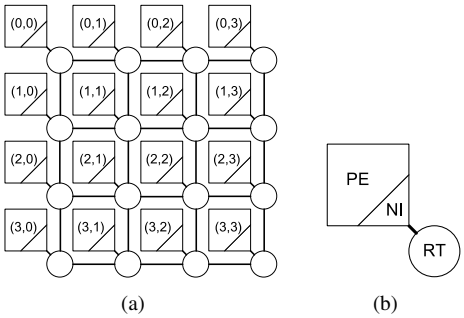

Fig. 1. (a) The GALS network architecture and (b) a network node

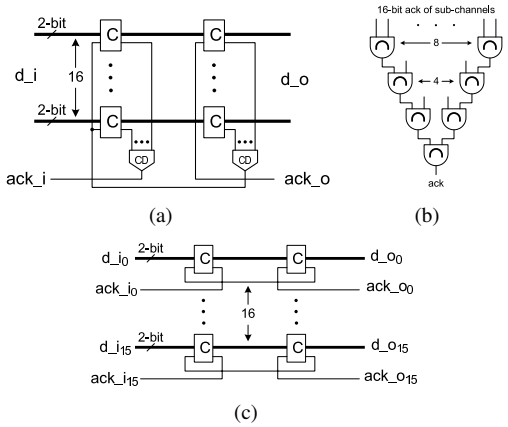

Fig. 2. (a) A 32-bit 1-of-4 pipeline, (b) the completion detection circuit and (c) the channel sliced pipeline

The data width of all ports is set to 32-bit to meet the throughput requirement for a normal multi-processor SoC application. A flit is also 32 bits and is transmitted in one cycle. A frame comprises a head flit, several data flits and a tail flit. The head flit contains a 1 byte of address, denoting the target node, and 3 bytes of data. The maximal size of a network is 16x16.

III. Way to Improve the Speed

A. Channel Slicing

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 [12], QNoC [13] and ASPIN [8]. The 4-phase dual-rail protocol has been used in ASPIN [8] and the 4-phase 1-of-4 protocol has been used in CHAIN [4], QoS [5] and ANoC [6]. Finally, m-of-n protocols have been used in SpiNNaker [7].

The 4-phase 1-of-4 protocol is preferred. Bundled-data protocols work under cautious timing constraints and the matched delay lines are vulnerable to process variation [14, 12]. M-of-n protocols transmit more data bits in one cycle than the 1-of-4 protocol but they need extra decoders and encoders [15]. Because the address in the head flit is analyzed by every router on the path, a decoder is added on each input port to translate the head flit, which introduces area overhead. The 4-phase 1-of-4 protocol is QDI, comparably area efficient than m-of-n protocols and more power efficient than the dual-rail protocol.

In all QDI routers, a wide channel is built by synchronizing multiple bit-level sub-channels [4, 5, 6, 7, 8], such as the 32-bit 1-of-4 channel shown in Fig. 2(a). The synchronized channel behaves similar to the pipeline formed by flip-flops in synchronous circuits. Techniques used in synchronous routers, such as the virtual channel, could be easily adopted. However, the completion detection (CD) circuit is a 16-input C-element tree, shown in Fig. 2(b). Assuming that all 2-input gates have the same latency and the C-element is a two level combinational logic, this completion detection circuit has 8 levels of logic. As the forward path of a basic 1-of-4 pipeline only has 4 levels of logic, the C-element tree accounts for 66% of the cycle period.

Synchronization is necessary for timing division multiple access (TDMA) technologies, such as the virtual channel flow control, but not for wormhole routers. According to the wormhole flow control method, the route is decided and reserved by the head flit and data flits simply follow the head flit. Since no frames could prevent data flits from following the head flit, no synchronization is needed. We propose to slice the synchronized channel into sub-channels, illustrated in Fig. 2(c), to allow independent data transmission on sub-channels and remove the time consuming C-element tree. Extra controllers are added to ensure that the head flit is successfully analyzed.

B. Lookahead Pipeline

Similar to synchronous circuits where throughput is constrained by the maximal latency between any two adjacent registers, the throughput of asynchronous circuits is constrained by the maximal cycle period of any two adjacent pipeline stages. The loop path of the two adjacent pipeline stages with the maximal cycle period is called the 'critical cycle' of the circuit.

Fig. 3 shows a part of the data path in a wormhole network. It is easy to observe that two loops could be the critical cycle: the loop around the long interconnects between routers and the loop that traverses the crossbar. A simple solution for the long interconnect between routers is to insert more pipeline stages in it. Many papers have concentrated on this long wire effect [16, 17, 18] and, thus, it is beyond the scope of this paper. The loop traversing the crossbar is the critical cycle.

The internal pipeline stages of routers are not necessarily strictly QDI. Utilizing some easily satisfiable timing constraints (see section IV-B), the cycle period of the critical cycle could be significantly reduced. We propose to use the lookahead pipeline [3] on the critical cycle to improve the overall throughput.

A QDI pipeline and a lookahead pipeline are shown in Fig. 4. Unlike the QDI pipeline, the ack line to stage N, in the lookahead pipeline, comes from the subsequent stage $N+1$ and the successor $N+2$. Stage N could receive a new data $D+1$ after D has been captured by stage $N+2$ instead of waiting D to be released by stage $N+1$ in the QDI pipeline. As reported, the dual-rail lookahead pipeline could reduce 27% cycle period

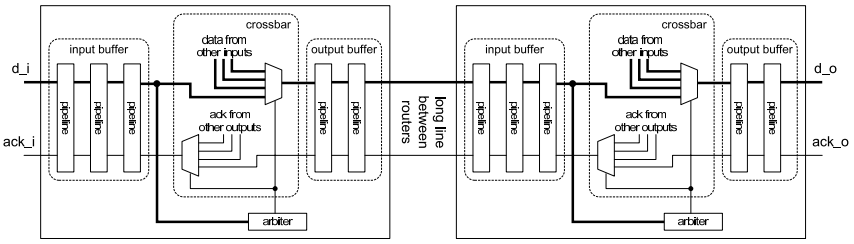

Fig. 3. The data path of wormhole networks

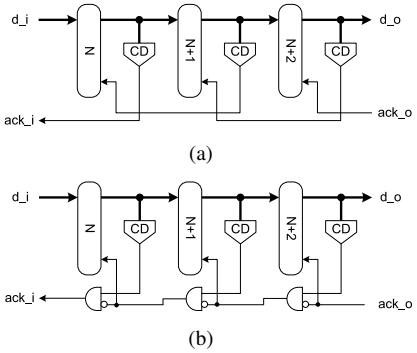

Fig. 4. (a) A QDI pipeline and (b) a Lookahead pipeline

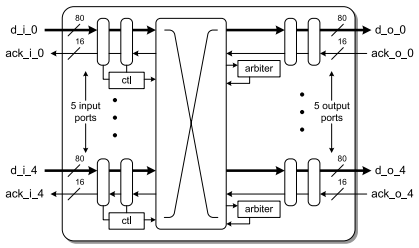

Fig. 5. Router structure

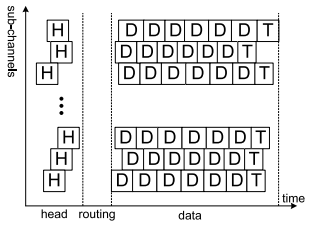

Fig. 6. The modified wormhole data flow

from the dual-rail QDI pipeline [3].

IV. Router Design

In the previous section, we have proposed to remove the synchronization in pipelines and use the lookahead pipeline style on the critical cycle. In this section, a wormhole router is implemented according to the proposed ideas.

A. Router Structure and Data Flow

Fig. 5 shows the internal structure of the proposed router. A router has five input and five output ports for four adjacent routers and the local network interface. A buffer with two pipeline stages is added on each input port and output port. Input buffers and output buffers are connected by a crossbar configured by the arbiter on each output port. Route decisions are made on each input buffer and routes are reserved by obtaining a grant from the corresponding arbiter on the output port. Since sub-channels run independently, they have their own ack wires. An end-of-frame (EOF) wire is also added to each sub-channel to identify the tail flit. As a result, one sub-channel has five data wires and one ack wire, the same as Chain [4]. A 32-bit channel has 16 sub-channels. Every port contains 80 data wires and 16 ack wires.

The basic wormhole data flow is slightly changed due to the removed synchronization. Fig. 6 shows the modified data flow. A flit is sliced into 16 parts and each of them is transmitted on a sub-channel. The head flit is firstly blocked in the first stage of the input buffer. Then the control logic analyzes the address in the head flit and makes a request to one of the arbiters. After the request is granted, a path is reserved in the crossbar and the frame is delivered by independent sub-channels. The crossbar is reset by the input buffer once all parts of the tail flit are delivered.

The head flit is blocked in the first stage of the input buffer instead of the last stage as in ASPIN [8] for two reasons: firstly, it reduces the fan-out of the second stage which is on the critical cycle; secondly the route decision procedure and the crossbar reset proceed in parallel.

Since the lookahead pipeline is utilized on the critical cycle, a two stage output buffer is added on each output port. As described in section III-B, the ack line of a lookahead pipeline stage is generated from the subsequent two pipeline stages; therefore, the ack line of the critical cycle is generated internally by the output buffer and the unknown latency on the paths between routers has no effect on the timing of the critical cycle. If the speed requirement is already met by using channel slicing alone, the output buffer could be removed to reduce the router latency.

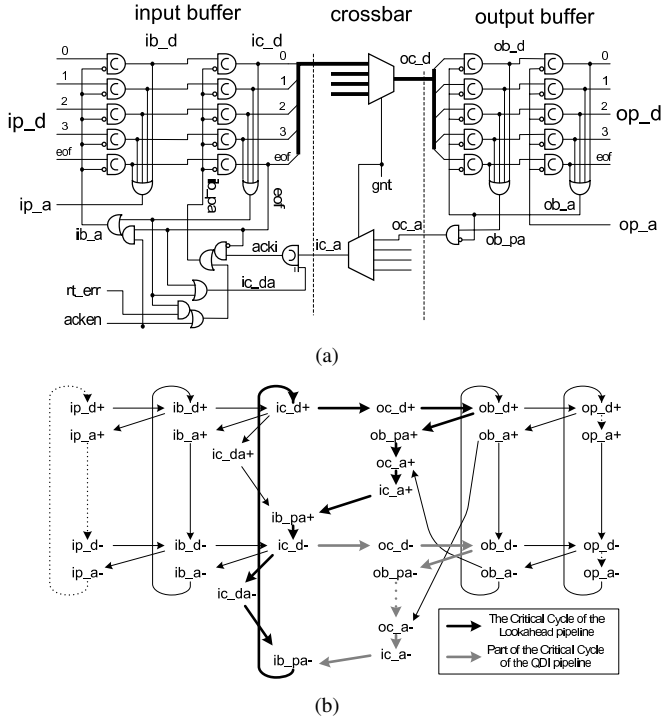

Fig. 7. (a) The data path of a sub-channel and (b) its STG

B. The Data Path of a Sub-channel

Fig. 7 illustrates the data path of a single sub-channel and its signal transition graph (STG). rt_err and acken are two signals driven by the extra controller added on each sub-channel (section IV-C). acken, the active low signal enabling the data path, is set low after a route request is initiated and it is driven to high to stall the data path when the tail flit is detected on ic_d. rt_err indicates incorrect route requests and is set high when a faulty frame is going to be dropped. gnt is the grant result from arbiters (section IV-D), which enables the MUXes and DEMUXes in the crossbar. Because the values of acken, rt_err and gnt are preserved during the whole data session, they are omitted in the STG.

A modified lookahead pipeline [3] is used to generate ib_pa. oc_a is the equivalent ack line generated by the lookahead pipeline. It is set after the ack signal ob_pa from the first stage in the output buffer and reset by ob_a from the successor stage. The pipeline stages of the original lookahead pipeline are dynamic logic, which are directly precharged by ack lines. However, the dynamic logic cannot be implemented by standard cells. Pipeline stages implemented by C-elements, used in this paper, reset after the release of the input data. If the data are not reset early enough and the new ack arrives too fast, the data path would be blocked. Therefore, a C2N element is added after ic_a to ensure ib_pa only drops when the data on ic_d is released (defer the new ack).

The STG of the lookahead pipeline is not speed-independent. If the transition from ob_d+ to oc_a+ is slow enough, oc_a could be reset even before it is firmly high. The length of the positive pulse on oc_a is ensured by timing constraints instead of STG. The dotted arrow from ob_pa- to oc_a- illustrates the timing assumptions present.

The critical cycle is highlighted by the dark bold line. Without using the lookahead pipeline, the critical cycle of normal the QDI pipeline traverses the crossbar four times (the grey bold line) because ic_d+ only occurs after the data on ob_d is released. Since the lookahead pipeline allows data to be captured in parallel with the reset of the next stage, the critical cycle traverses the crossbar twice. The cycle period is reduced.

Two timing constraints must be satisfied for the correct data path operation.

Ack setup time: data on ic_d is cleared by ib_pa+. Thus, the positive pulse on ic_a must remain long enough to make it captured by the C2N gate. This constraint includes two timing relations:

$$t_{\mathrm{ic\_d+}\rightarrow\mathrm{ic\_da+}} < t_{\mathrm{ic\_d+}\rightarrow\mathrm{ic\_a+}} \tag{1}$$

$$t_{\mathrm{ob\_d+}\rightarrow\mathrm{ic\_a-}} - t_{\mathrm{ob\_d+}\rightarrow\mathrm{ic\_a+}} > t_{\mathrm{C2N\_setup}} \tag{2}$$

Equation (1) ensures that the C2N element is ready to capture the ack pulse on ic_a before its arrival. As the transition from ic_d+ to ic_a+ traverses the crossbar, it is always satisfied. Equation (2) requires the length of the pulse on ic_a is long enough to stabilize the feedback loop in the C2N element. It could be easily met by constraining the minimal delay of the transition from ob_d+ to op_d+ and the throughput is not affected because this transition is outside the critical cycle.

Data override: The new data should be securely captured after the previous data is cleared. In the proposed router, ob_d is the only pipeline stage not ensured by STG. To avoid the data override on ob_d,

$$t_{ic\_d-\rightarrow oc\_d+} - t_{ic\_d-\rightarrow oc\_d-} > t_{C2\_setup} \tag{3}$$

This constraint is already satisfied by hardware. Both transitions in (3) share the path from ic_d to ob_d. Suppose the positive and negative transitions on this path are around the same speed, the left side of Equation (3) is the length of the negative pulse on ic_d. Since the minimal length of this pulse is half of the period of the fastest 1-of-4 pipeline, it is normally larger than the setup time of a C-element.

C. Channel Control

Although sub-channels run in parallel during the data session, they stall after the tail flit to keep the next head flit in the first pipeline stage in the input buffer. An input buffer has one route decision controller and several sub-channel controllers, one for each sub-channel. For an incoming frame, the route decision controller enables the route decision procedure. Once a route request is initiated, sub-channel controllers enable their data paths.

Fig. 8 demonstrates the internal structure of the route decision controller and its STG. The route decision procedure is always enabled through rt_en+ after a frame is transmitted. A route decision could be a possible route request (rt_dec+) or a faulty request (rt_err+). The frame generating a faulty request will be dropped. After the route request is made, the route decision procedure is disabled until the frame is transmitted, denoted by ch_fin+ on all sub-channels.

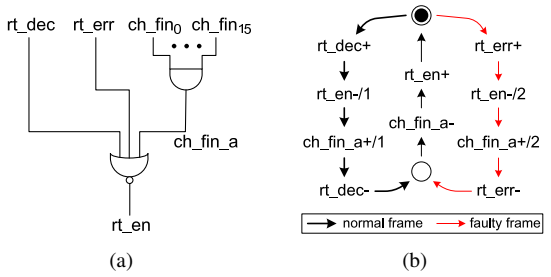

Fig. 8. (a) A route decision controller and (b) its STG

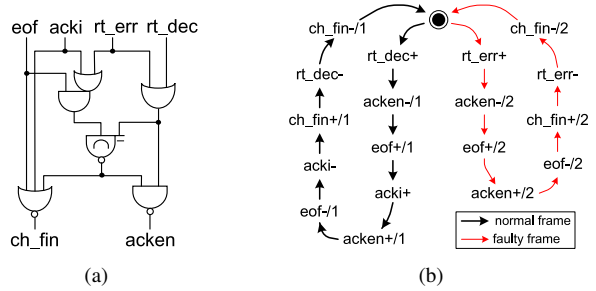

Fig. 9. (a) A sub-channel controller and (b) its STG

Fig. 9 shows a sub-channel controller and its STG. A data session begins after a route request is made. A faulty frame is dropped by connecting the ack line generated from ic_d directly to itself, enabled by rt_err in Fig. 7(a). Note that the ack line connected back is generated from data bits but not the EOF bit to guarantee that the EOF bit is always detected by the sub-channel controller. When the tail flit arrives, it is dropped by acken+ and then the sub-channel stalls until the next data session. For normal frames, the ack line acki from output buffers is used. As the output of the C2N element added on ic_a in Fig. 7(a), acki only drops when the data on ic_d is released.

D. Routing and Arbitration

As an example, Fig. 10 shows the route decision circuit in the south input buffer and the connected arbiter on the east port. Enabled by rt_en, the 8-bit address (16-bit in 1-of-4 code) blocked in the first pipeline stage enters comparators after the second pipeline stage is cleared (ib_a is low). The route request is captured by C2P elements enabled by ch_fin_a-. One-hot coded, the route request drives rt_dec or rt_err to '1', which then disables the route decision procedure and starts the data session. C2P elements hold the value during the whole data session. The south input buffer could not be connected with the south output buffer, therefore, the corresponding route request is connected to rt_err.

Valid route requests are sent to arbiters. Since a maximum of four input buffer could request to one output port concurrently, the multi-way MUTEX arbiter [19], shown in Fig. 10, is faster and smaller than other arbiter styles [20, 21, 19]. The successful request is granted by one of the four gnt outputs.

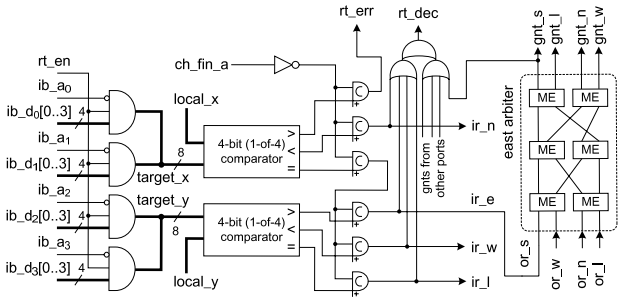

Fig. 10. The routing decision circuit and the arbiter

V. Performance

A. Physical Implementation

The router has been implemented using the Faraday 0.13 μm standard cell library based on the UMC 0.13 μm technology. Route decision controllers and sub-channel controllers are speed independent circuits generated from their STGs using Petrify [22] and other parts are manually written in Verilog HDL.

The area after synthesis is around 14.3K gates (0.057 mm^2). The final router is placed and routed on a 0.3x0.3 mm block using 5 metal layers. The speed simulation is back-annotated with the RC extraction from the layout and run under the typical corner (25 °C, 1.2 V). The cycle period for data flits is 1.7 ns, providing maximal 2.35 GByte/s throughput on a single port. The average latency of a data flit is also 1.7 ns. For the head flit, the routing decision and the arbitration procedure consume about 0.8 ns without contention.

B. Effect of Channel Slicing and the Lookahead Pipeline

Channel slicing and the lookahead pipeline are the two major contributions of this paper. Removing the C-element tree in the completion detection circuit in Fig. 2(a), channel slicing (ChSlice) splits a synchronized asynchronous channel into multiple independent sub-channels, which reduces the cycle period. However, the increased wire count and extra sub-channel controllers increase area. It is important to evaluate the area overhead of ChSlice against its speed benefit. The lookahead (LH) pipeline reduces cycle period through the early evaluation protocol. Although all constraints required by LH are satisfiable, they would make the data path vulnerable to the extreme process variation. It is interesting to evaluate the performance of a router only with ChSlice.

To answer these questions, a router without ChSlice or LH (the router using QDI synchronized channels) and a router only with ChSlice are implemented. Table I shows the area after synthesis and Table II illustrates the speed performance after RC extraction.

ChSlice significantly increases the area of input buffers and crossbars but output buffers. ChSlice increases the wire count of data paths but also removes the C-element tree. The C-element tree in the router without ChSlice or LH utilizes 15 C-elements. ChSlice adds 16 C-elements for EOF bits and increases the fan-in of the OR gate in the completion detection

TABLE I
AREA OVERHEAD OF CHSLICE AND LH

Block	ChSlice & LH	ChSlice	No ChSlice/LH
Input Buffers	6.2K	5.8K	4.3K
Output Buffers	4.5K	4.5K	4.4K
Crossbar	3.3K	3.2K	2.4K
Total	14.5K	13.9K	11.3K

TABLE II
SPEED IMPROVEMENT OF CHSLICE AND LH

	ChSlice & LH	ChSlice	No ChSlice/LH
Period	1.7 ns	2.2 ns	2.9 ns
Latency	1.7 ns	2.1 ns	2.8 ns
Route Overhead	0.8 ns	0.8 ns	0.8 ns

circuit from four to five. Because the removed C-element tree compensates the area of the extra C-elements introduced by ChSlice, ChSlice only adds one extra C-element to each output buffer and increases the fan-in of OR gates, which explains the slightly changed area of the output buffers. The area of the crossbar increases because it is linear to the wire count. The area of the input buffers grows due to the extra sub-channel controllers. The LH technique only increases the area of the input buffers significantly. Shown in Fig. 7(a), the added C2N element sites on the critical cycle. They are severely optimized with larger driven strength and buffer insertion during synthesis. As a result, ChSlice introduces 23.0% area overhead and the LH pipeline causes further 5.3% overhead.

ChSlice and LH reduce the cycle period by 24.1% and 17.2% respectively, as shown in Table II. ChSlice and LH reduce 41.4% cycle period (70.6% improvement in peak throughput) with 28.3% area overhead, compared with the router without them.

C. Compare with Other Asynchronous Routers

Table III compares the performance of asynchronous routers published in recent years.

The bundled-data protocol has been used by MANGO, ASPIN and QNoC. Instead of detecting the implicit request from data, the bundled-data pipeline stages use matched delay lines to defer the exclusive requst until the arrival of data. This non-detection structure makes bundled-data pipelines fast and most high speed asynchronous FIFOs are built by them [23, 24, 25], including the custom designed FIFOs in the ASPIN router [8]. However, the matched delay lines intensively rely on the accurate timing estimation, which is unreliable in the presence of process variation. As a result, although our router is slower than MANGO and ASPIN, it is immune to process variation.

Several designs have utilized special cell libraries. ANoC has used the augmented cell library from TIMA [26] for C-elements and MUTEXes. ASPIN has also used a set of special designed asynchronous cells for the SXLIB cell library [27]. Our router is a pure standard cell implementation; therefore, it could be easily adapted to other technologies or shipped out in the form of a soft IP. On the other hand, the speed could be further improved by using those special cell libraries.

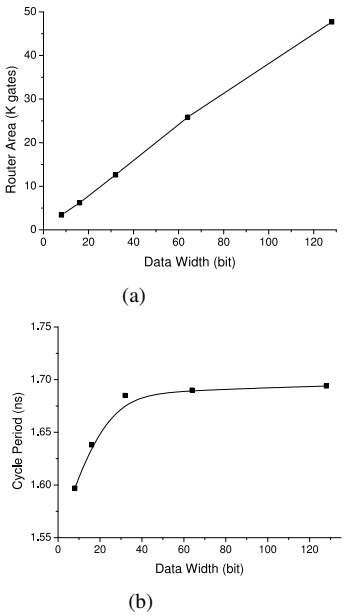

Fig. 11. (a) Area and (b) cycle period under different data width

D. Effect of Data Width

An extra and important advantage of ChSlice is that the cycle period does not increase with the width of data paths. For normal QDI pipelines, increasing the data width means more sub-channels are synchronized by the C-element tree leading to a larger speed penalty. The bundled-data router in QNoC also reported similar effect (0.2% per bit degradation [13]).

However, for the routers using ChSlice, sub-channels run independently during the data session. Increasing the number of sub-channels has little impact on the cycle period of a single sub-channel. Several routers with different data widths have been implemented and Fig. 11 shows the results after synthesis. Although the area increases linearly with the data width, the cycle period remains around 1.7 ns.

VI. CONCLUSION

In this paper, a low latency asynchronous router has been implemented. The router utilizes two novel techniques: channel slicing and the lookahead pipeline.

Channel slicing removes the C-element tree in the completion detection circuit of QDI pipelines. This removal reduces the cycle period and make sub-channels run in parallel during the data session. The router implementation shows that channel slicing reduces the cycle period by 24.1% for a 32-bit wormhole router with 23.0% area overhead. Due to the parallelization of sub-channels, the router does not suffer the degraded cycle period as reported by QNoC [13].

The lookahead pipeline is a fast pipeline style allowing early acknowledge generation, proposed by Montek [3]. For a wormhole router, the peak throughput is determined by the critical cycle. We propose to use the lookahead pipeline on the critical cycle to increase throughput. Implementation results show that it reduces the cycle period by 17.2% with 5.3% area overhead.

TABLE III
ASYNCHRONOUS ROUTER COMPARISON

Router	Period	Latency	Tech	Library & Layout	Protocol
MANGO [12]	1.26 ns	unknown	0.12 μm	unknown	bundled-data
ANoC [6]	4 ns	2 ns	0.13 μm	augmented cell lib	1-of-4
QNoC [13]	4.8 ns	10 ns	0.18 μm	standard cell lib	bundled-data
ASPIN [8]	0.88 ns	1.53 ns	90 nm	partial customized	dual rail & bundled-data
Our Router	1.7 ns	1.7 ns	0.13 μm	standard cell lib	1-of-4 & Lookahead

The final router using both channel slicing and the lookahead pipeline has been implemented on a 0.3x0.3 mm block using the Faraday 0.13 μm standard cell technology. The synthesis result is around 14.5K gates. Simulations are back-annotated with RC extraction and run at the typical corner. The cycle period is around 1.7 ns providing 2.35 GByte/sec throughput on each port.

Since channel slicing and the lookahead pipeline can reduce the cycle period of asynchronous pipelines, they could be used in the design of network interfaces. However, channel slicing removes the synchronization between sub-cahnnels which hinders the use of any timing division multiple access techniques, such as the virtual channel flow control. The spatial division multiplex techniques [28] could be adopted to provide similar functions.

ACKNOWLEDGEMENTS

The authors would like to thank the financial support from EPSRC under grant EP/E06065X/1.

REFERENCES

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

[2] A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist, "Lowering power consumption in clock by using globally asynchronous locally synchronous design style," in *Proc. of DAC*, 1999, pp. 873–878.

[3] M. Singh and S. M. Nowick, "The design of high-performance dynamic asynchronous pipelines: lookahead style," *IEEE Transactions on VLSI Systems*, vol. 15, no. 11, pp. 1256–1269, November 2007.

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

[5] T. Felicijan and S. B. Furber, "An asynchronous on-chip network router with quality-of-service (QoS) support," in *Proc. of SOCC*, Sept. 2004, pp. 274–277.

[6] E. Beigné, 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.

[7] 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.

[8] A. Sheibanyrad, "Asynchronous implementation of a distributed network-on-chip," Ph.D. dissertation, University of Pierre et Marie Curie, 2008.

[9] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø, "An OCP compliant network adapter for GALS-based SoC design using the mango network-on-chip," in *Proceedings of International Symposium on System-on-Chip*, 2005, pp. 171–174.

[10] A. Sheibanyrad and A. Greiner, "Two efficient synchronous asynchronous converters well-suited for networks-on-chip in GALS architectures," *Integration, the VLSI Journal*, vol. 41, no. 1, pp. 17–26, 2008.

[11] Y. Thonnart, E. Beigné, and P. Vivet, "Design and implementation of a GALS adapter for ANoC based architectures," in *Proc. of ASYNC*, May 2009, pp. 13–22.

[12] 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.

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

[14] B. Liu, "Robust differential asynchronous nanoelectronic circuits," in *Proc. of ISQED*, March 2009, pp. 97–102.

[15] 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.

[16] S. Hollis and S. W. Moore, "RasP: an area-efficient, on-chip network," in *Proc. of ICCD*, October 2006, pp. 63–69.

[17] R. R. Dobkin, Y. Perelman, T. Liran, R. Ginosar, and A. Kolodny, "High rate wave-pipelined asynchronous on-chip bit-serial data link," in *Proc. of ASYNC*, 2007, pp. 3–14.

[18] C. D'Alessandro, A. Mokhov, A. Bystrov, and A. Yakovlev, "Delay/phase regeneration circuits," in *Proc. of ASYNC*, 2007, pp. 105–116.

[19] D. J. Kinniment, *Synchronization and Arbitration in Digital Systems*. John Wiley & Sons Inc., 2007.

[20] K. S. Low and A. Yakovlev, "Token ring arbiters: An exercise in asynchronous logic design with Petri nets," Newcastle University, Tech. Rep., 1995.

[21] M. B. Josephs and J. T. Yantchev, "CMOS design of the tree arbiter element," *IEEE Transactions on VLSI*, vol. 4, no. 4, pp. 472–476, Dec 1996.

[22] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, vol. E80-D, no. 3, pp. 315–325, 1997.

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

[24] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in *Proc. of ASYNC*, 2001, pp. 46–53.

[25] P. Wielage, E. J. Marinissen, M. Altheimer, and C. Wouters, "Design and DfT of a high-speed area-efficient embedded asynchronous FIFO," in *Proc. of DATE*, 2007, pp. 853–858.

[26] P. Maurine, J. Rigaud, F. Bouesse, G. Sicard, and M. Renaudin, "Static implementation of QDI asynchronous primitives," in *Proc. of PATMOS*, 2003, pp. 181–191.

[27] A. Greiner and F. Pcheux, "ALLIANCE: A complete set of CAD tools for teaching VLSI design," in *Proc. of the 3rd Eurochip Workshop on VLSI Design Training*, 1992, pp. 230–237.

[28] A. Leroy, D. Milojevic, D. Verkest, F. Robert, and F. Catthoor, "Concepts and implementation of spatial division multiplexing for guaranteed throughput in networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1182–1195, September 2008.