Improving the Throughput of Asynchronous On-chip Networks with SDM

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

***Abstract*—Asynchronous quasi-delay-insensitive (QDI) NoCs have several advantages over their clocked counterparts. Virtual channel (VC) is the most utilized flow control method in asynchronous routers but spatial division multiplexing (SDM) achieves better throughput performance for best effort traffic than VC. A novel asynchronous SDM router architecture is presented. Area and latency models are provided to analyse the network performance of all router architectures including wormhole, virtual channel and SDM. Network performance shows that SDM routers outperform VC routers in both throughput and area overhead.**

I. Introduction

Flow control is one of the major NoC research issues. Most router designs are in favour of the virtual channel (VC) flow control method [1]. A VC is a distinct set of buffers storing the data of an individual frame. With the help of these VCs, a port can deliver flits of multiple frames in a time division manner as long as each frame is allocated with a VC. When a frame is temporarily stalled in a router, the virtual channel flow control method improves throughput by allowing other frames to utilize the link.

Although the virtual channel flow control method improves the overall throughput significantly, employing it in asynchronous routers leads to extra area and speed overhead: (1) Every VC is a separated set of buffers. VC routers normally need deep buffers to compensate the credit loop latency [2] and maximize throughput. As a result, buffers consume most of the router area and the area overhead is significant. (2) Network throughput is determined by the maximal loop latency on data paths. In VC routers, the crossbar is reconfigured in every flit. This frequent reconfiguration introduces an extra configuration latency in every cycle. The throughput is thus compromised.

As an alternative way to improve throughput, the spatial division multiplexing (SDM) flow control method physically divides every link and buffer into several virtual circuits [3], [4]. Each virtual circuit exclusively occupies a portion of the bandwidth and runs independently using the basic wormhole flow control method. Hence, only a portion of the buffer resources halt when a frame is blocked. Results in latter sections will show that the SDM flow control method improves throughput significantly.

More importantly, an asynchronous SDM router suffers less than an asynchronous VC router. The major area overhead of SDM routers is that the size of the crossbar is proportional to the number of virtual circuits. Since the crossbar normally consumes less area than buffers, an SDM router with M virtual circuits is smaller than a VC router with M VCs. For speed performance, no extra configuration latency is introduced because virtual circuits do not share resource until a whole frame is delivered.

In this paper, we propose and implement the first asynchronous spatial division multiplexing router for best-effort on-chip networks. Area and latency estimation models for the wormhole, the virtual channel and the spatial division multiplexing flow control methods are provided. We build up latency accurate SystemC models to analyse all these flow control methods in an 8x8 mesh network. All the simulation results show that SDM routers outperform VC routers in both area and throughput performance.

The remainder of this paper is organized as follows: Section II demonstrates the hardware architecture of the proposed asynchronous SDM router. Section III provides area and latency estimation models and verifies them with the results from practical implementations. Based on these models, section IV reveals the network performance of using various flow control methods with different configurations. Finally the paper is concluded in Section V.

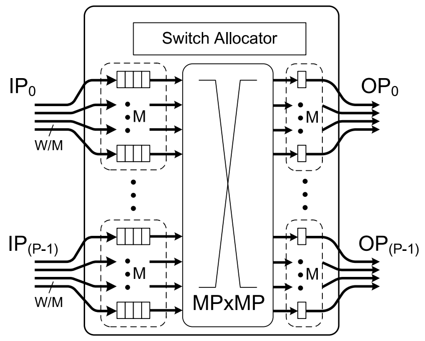


Fig. 1: An SDM router with P ports of width W bits and M virtual circuits in each port

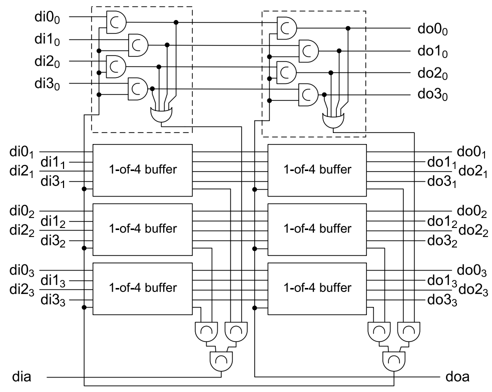


Fig. 2: Two 8-bit QDI buffer stages

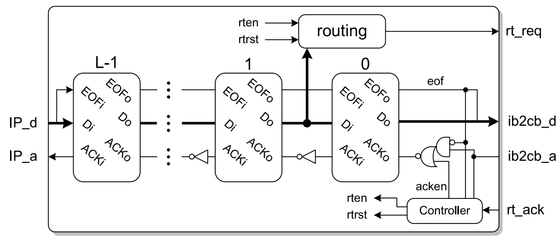


Fig. 3: Structure of an input buffer

II. Implementation

A. Overall Architecture

An asynchronous SDM router is illustrated in Fig. 1. The router comprises P input buffers, P output buffers, a crossbar and a switch allocator. The bandwidth for each port is W bits. Inside the router, every buffer is physically divided into M virtual circuits. As a result, the crossbar is an $MP \times MP$ crossbar and every port has a bandwidth of W/M bits. To allocate such a crossbar, the switch allocator uses an arbitration scheme of $MP \times MP$. Theoretically, output buffers are unnecessary but they are crucial to throughput performance. Most asynchronous routers do have at least one stage of them [5]–[9].

The 4-phase 1-of-4 protocol is used in our router implementations. A wide QDI buffer stage is normally formed by synchronizing multiple bit-level buffer stages. Fig. 2 shows an example of two 8-bit buffer stages. Each 1-of-4 buffer latches two data bits. All the four bit-level buffers share the common ACK line (dia and doa) generated by a C-element tree. For a general wide buffer with W bits, $W/2$ 1-of-4 buffers and $W/2-1$ C-elements are required to latch data and form the C-element tree. In router designs, an end-of-frame (EOF) bit is added into data paths to identify tail flits.

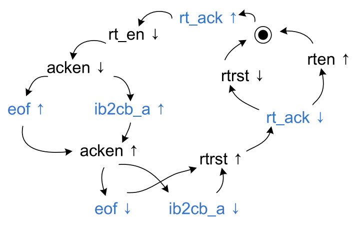
B. Input buffers

Every input virtual circuit is a fully functional input buffer as in a wormhole router. As shown in Fig. 3, it contains L stages of buffers, a buffer controller and a routing calculation circuit.

The ACK line of the 0th buffer stage is controlled by the buffer controller through the ACK enable signal acken. At the beginning, the 0th buffer stage is stalled. Thus the head flit of the coming frame waits in the 1st buffer stage. Meanwhile, the routing enable signal rten is also set to allow the routing calculation circuit to analyse the head flit and generate a routing request to the switch allocator through rt_req. Once a positive acknowledgement is received from rt_ack, the buffer controller withdraws acken to open the data path.

A frame is always terminated with a tail flit, in which only the EOF bit is set. When the tail flit has been latched in the 0th stage, direct acknowledgement from ib2cb_a is masked by the EOF bit. The 0th stage is, again, under the full control of the buffer controller. When ib2cb_a and eof are both high, denoting that the tail flit is successfully captured by the output buffer, the buffer controller clears the 0th buffer stage and disables it by depositing acken to high. Simultaneously, rten is also withdrawn to release the reserved path in the crossbar. Finally, the routing calculation circuit is enabled again after the path is released (indicated by rt_ack). The input buffer then waits for the next frame. Fig. 4 illustrates the signal transition graph (STG) of the buffer controller and the circuit generated using Petrify [10].

Fig. 5 shows a routing calculation circuit. addrx and addry are the target addresses extracted from the head flit. Comparison results are captured in and translated into a routing request by the following asymmetric C-elements. The routing request is released when rtrst is set to high.


(a) STG

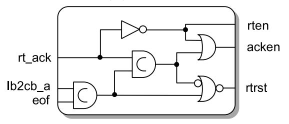

(b) Schematic

Fig. 4: Buffer controller

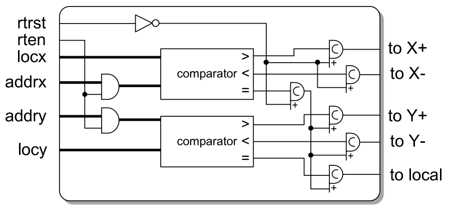
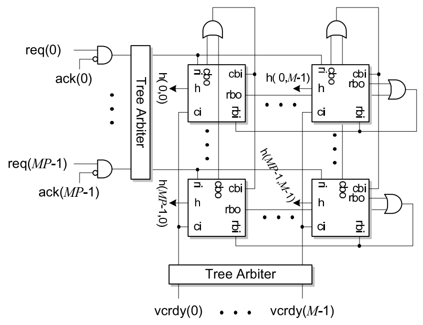


Fig. 5: Routing calculation circuit


(a) Allocation matrix

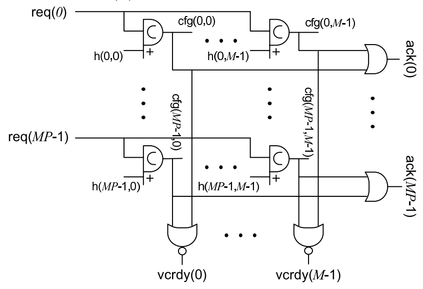

(b) CFG Capture matrix

Fig. 6: Switch allocator

C. Switch allocator

Although there is only one switch allocator shown in Fig. 1, the practical router contains P switch allocators, one per output port. The switch allocator receives requests from all input virtual circuits and allocates the idle virtual circuits in its output port to these requests. Thus every switch allocator has an arbitration scheme of $MP \times M$. Traditional asynchronous arbiters, such as the MUTEX-NET arbiter [8], [11], the tree arbiter [12] and the ring arbiter [13] are capable of allocating only one resource to multiple clients. The multi-resource arbiter [14]–[16] is the only allocator that can fairly allocate multiple resources to multiple clients.

Fig. 6 depicts the internal structure of a switch allocator. It comprises two matrices: the allocation matrix and the configuration capture matrix. The allocation matrix shown in Fig. 6a is a tile-based multi-resource arbiter [15]. There are PM rows and M columns in the matrix where each row is driven by the request from an input virtual circuit and each column is driven by the status of an output virtual circuit in the local output port. The tile matrix is capable of matching one row to one column once upon a time. When multiple requests arrive simultaneously, these requests are served sequentially. To reduce the allocation latency, the matched row and column pair must be captured and then withdrawn as soon as possible to allow other pairs to be matched. The matched result h[r][c] is, therefore, captured by the configuration capture matrix. Consequently, ack[r] and vcrdy[c] signals are deposited and toggled to release the request lines. A tree arbiter [12] is added on both rows and columns to guarantee the one-hot condition in both of them.

Fig. 6b shows the configuration capture matrix. It contains a matrix of asymmetric C-elements. Each of them captures a match result h[r][c] and generates the configuration signal cfg[r][c] for the crossbar. They are enabled by the requests from input virtual circuits. An OR gate tree on each row generates ack[r] to input virtual circuits. A NOR gate tree on each column generates the status of local output virtual circuits vcrdy[c]. As the allocation matrix ensures that only one h[r][c] is fired in each row and column, the cfg[r][c] signal matrix is also one-hot in all rows and columns. Once the positive pulse on h[r][c] is captured, it is withdrawn immediately to allow another match to be made. The configuration bit cfg[r][c] is finally released when the request req[r] is withdrawn by the input virtual circuit.

D. Other components

An output virtual circuit contains no control logic but only one stage of buffer. Links in the crossbar are released by input virtual circuits once a tail flit is received by the output buffer. The crossbar is formed

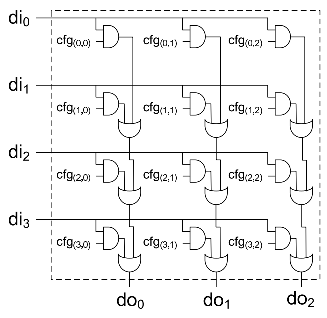


Fig. 7: A 4x3 crossbar with 1-bit bandwidth

by AND gates and OR gates, such as the 4x3 crossbar shown in Fig. 7.

III. Hardware Performance

In this section, a basic wormhole router and an SDM router are implemented using the Faraday standard cell library based on the UMC 0.13 μm technology and standard synthesis tools. Every router has five ports for the normal mesh topology. The bandwidth of each port is 32 bits which is common in MPSoC systems. A head flit contains an 8-bit address field and three bytes of data. In SDM routers, every buffer is divided into four virtual circuits. Two buffer stages are implemented in all input buffers and one stage of output buffer is utilized.

A. Area consumption

Before revealing the practical area consumption, area models are provided to estimate the area overhead of using various flow control methods with different parameters.

For any router architectures, the total area is expressed as:

$$A = P \cdot (A_{IB} + A_{OB}) + A_{CB} + A_A \tag{1}$$

where P is the port number. A_{IB}, A_{OB}, A_{CB} and A_A are the area of an input buffer, an output buffer, the crossbar and all allocators.

An input buffer in a wormhole router (WH) contains L buffer stages, a routing calculation circuit and a buffer controller. Denoting the bandwidth as W bits, each buffer stage has $2W+1$ C-elements to store data and EOF, and $0.5W-1$ C-elements to generate the common ACK signal. Therefore, the area of one input buffer is:

$$A_{IB,WH} = L \cdot (2.5W \cdot A_C + A_{EOF}) + A_{RC} + A_{CTL} \tag{2}$$

where A_C, A_{EOF},A_{RC} and A_{CTL} are the area of a single C-element, the extra logic introduced by the EOF bit, the routing calculation circuit and the buffer controller. As an output buffer is simply one stage of buffer, its area can be calculated as:

$$A_{OB,WH} = 2.5WA_C + A_{EOF} \tag{3}$$

An SDM router contains M virtual circuits with a bandwidth of W/M bits. Every virtual circuit is fully functional as the input buffer in a wormhole router. The output buffer is also divided into M virtual circuits. The area of an input buffer and an output buffer in an SDM router is as follows:

$$A_{IB,SDM} = M \cdot [L \cdot (2.5\frac{W}{M} \cdot A_C + A_{EOF}) + A_{RC} + A_{CTL}] \tag{4}$$

$$A_{OB,SDM} = 2.5WA_C + MA_{EOF} \tag{5}$$

The area of the crossbar is determined by the number of ports and the wire count of each port. The crossbar inside a wormhole router has P ports while the one in an SDM router has MP ports. Using the crossbar structure shown in Fig. 7, the area of the crossbars in all routers is as follows:

$$A_{CB,WH} = (2W + 2)(2P^2 - P)A_g \tag{6}$$

$$A_{CB,SDM} = (2\frac{W}{M} + 2)(2M^2P^2 - MP)A_g \tag{7}$$

where A_g is the equivalent area of a 2-input standard gate.

The area of allocators is difficult to estimate. It depends on the arbitration scheme and the internal structure. In this paper we assume the multi-resource arbiter is used in all allocators. Therefore, the area is approximately linear with the arbitration scheme. The area of the switch allocators in all routers can be estimated as:

$$A_{A,WH} = P^2 A_{arb} \tag{8}$$

$$A_{A,SDM} = M^2 P^2 A_{arb} \tag{9}$$

where A_{arb} is the equivalent area overhead of a single arbitration point in the arbitration scheme.

Using the area consumption from practical implementations, we have extracted all parameters as follows (in unit of μm^2):

$$A_C = 14.7 \qquad A_{EOF} = 11 \qquad A_{RC} = 440$$
$$A_{CTL} = 45 \qquad A_g = 2.45 \qquad A_{arb} = 86$$

The practical area consumption and the estimation error rates of the proposed models are illustrated in Table I. The only significant error occurs on the switch allocator in the wormhole router. The switch allocator in a P ports wormhole router is divided into P separated allocators with an arbitration scheme of $P \times 1$. In this case, it is unnecessary to use the multi-resource arbiter

TABLE I: Area consumption (μm^2)

	WH	err(%)	SDM	err(%)
Input Buffers	14,303	0.0	21,995	-0.4
Output Buffers	5,935	0.0	6,000	1.7
Crossbar	4,356	0.0	21,744	-0.2
Switch Allocator	772	78.2	22,208	-0.9
Total	25,366	2.4	71,956	-0.3

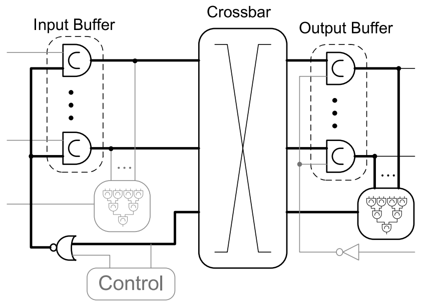


Fig. 8: The critical cycle in asynchronous routers

and the multi-way MUTEX arbiter [11] is utilized to reduce arbitration latency and area overhead. As shown in Table I, our models have successfully estimated the area of all router components within an maximal error rate of 3.4% excepted for the switch allocator in the wormhole router.

B. Latency analysis

For an asynchronous pipeline, critical cycle is the loop path between two continuous pipeline stages which has the maximal loop latency in all pairs of continuous stages. The throughput of an asynchronous pipeline is determined by the loop latency of its critical cycles just as the frequency of a synchronous system is determined by the latency of its critical paths. It is easy find out that the critical cycles for all router architectures in this paper are the loop paths traversing the crossbar.

A schematic view of the critical cycle is shown in Fig. 8 where the loop path is highlighted in bold lines. To transmit one data flit on a 4-phase pipeline, a series of transactions traverse the loop path twice: one for data transmission and one of buffer reset. The control circuit in input buffers is added on the ACK line. To simplify the latency calculation, we assume the propagation latencies for positive and negative transitions are the same in all gates.

The loop latency T of the critical cycle can be estimated as:

$$T = 4t_C + 4t_{CB} + 2t_{CD} + 2t_{AD} + t_{CTL} \tag{10}$$

where t_C is the propagation latency of a single C-element on data paths, t_{CB} is the propagation latency of the crossbar, t_{CD} is the propagation latency of the completion detection circuit which includes the C-element tree, t_{AD} is the ACK driver latency (the propagation latency of the NOR gate in Fig. 8) and t_{CTL} is the possible latency caused by the controller.

We use linear models to approximate gate latencies. The C-elements on data paths are connected with the crossbar; therefore, t_C is linear with the port number of the crossbar.

$$t_C = \begin{cases} l_C + k_C(P+1) & \text{wormhole,} \\ l_C + k_C(MP+1) & \text{SDM.} \end{cases} \tag{11}$$

where l_c is the latency of a C-element with zero load and k_c is the fanout factor (the extra latency introduced by every new fanout). For the same reason, the ACK driver latency is linear with the wire count of the data path.

$$t_{AD} = \begin{cases} l_{AD} + k_{AD}(2W+1) & \text{wormhole,} \\ l_{AD} + k_{AD}(2W/M+1) & \text{SDM.} \end{cases} \tag{12}$$

where l_{AD} and k_{AD} are the latency of the ACK driver without load and the fanout factor for t_{AD}.

The crossbar has two levels of gates: an AND gate matrix and OR gate trees. If all gates are implemented in 2-input standard gates, the depth of the OR gate tree is determined by the number of input ports.

$$t_{CB} = \begin{cases} l_{CB} + k_{CB} \cdot log_2(P) & \text{wormhole,} \\ l_{CB} + k_{CB} \cdot log_2(MP) & \text{SDM.} \end{cases} \tag{13}$$

where l_{CB} and k_{CB} are the propagation latency of an AND gate and an OR gate respectively.

The latency of the completion detection circuit is complicated. It contains a C-element tree. The fanout of the final common ACK signal depends on the number of input ports in the crossbar. The latency estimation must consider the impact by both factors.

$$t_{CD} = \begin{cases} l_{CD} + l_C \cdot log_2(W/2) + k_{CD} \cdot P & \text{wormhole,} \\ l_{CD} + l_C \cdot log_2(\frac{W}{2M}) + k_{CD} \cdot MP & \text{SDM.} \end{cases} \tag{14}$$

where l_{CD} and k_{CD} is the zero load latency of a completion detection circuit without any C-elements (the latency of two OR gates) and the fanout factor for t_{CD}. For wormhole and SDM routers, the buffer controller halts a data path once per frame; therefore, the extra control latency for these routers is zero during data transmission.

We have extracted all parameters through the post-synthesis simulations. The extracted parameters are listed as follows (in unit of ns):

TABLE II: Speed performance (ns)

	WH	err	SDM	err(%)
cycle period	4.25	2.6	4.15	-3.4
router latency	2.29		2.49	
routing calculation	0.44		0.51	
switch allocation	0.78		3.21	
t_C	0.22	-9.1	0.34	-5.9
t_{CB}	0.16	1.3	0.26	-3.8
t_{CD}	0.79	7.6	0.57	4.2
t_{AD}	0.57	6.1	0.27	-0.4

$l_C = 0.15$	$k_C = 0.01$	$l_{CB} = 0.074$
$k_{CB} = 0.044$	$l_{CD} = 0.23$	$k_{CD} = 0.004$
$l_{AD} = 0.15$	$k_{AD} = 0.007$	

Table II reveals the practical speed performance of all routers from post-synthesis simulations. "Cycle period" is the loop latency of the critical cycle. Using the latency estimation models and the parameters above, Table II also shows the estimation error rates. The latency estimation has larger error rate than the area estimation for two reasons: the practical gate latency model is not linear; the propagation time of a gate is also affected by the input transition time which is difficult to measure statically.

If routers are implemented synchronously, an SDM router is slower than a wormhole router; however, the asynchronous SDM router shows a smaller cycle period than the asynchronous wormhole router. Undoubtedly, SDM increases the port number of the crossbar, which leads to longer C-element propagation delay, crossbar traversing latency and t_{CD}. Meanwhile, every virtual circuit has only a portion of the total bandwidth. The depth of the C-element tree in the completion detection circuit is reduced, which reduces t_{CD}. The ACK driving latency t_{AD} is also reduced as less C-elements are driven by one ACK line. The latency reduction has compensated the extra latency introduced by the crossbar.

C. VC router

As depicted in Fig. 9, an asynchronous VC router comprises P input buffers, an $MP \times P$ crossbar, P output buffers, a VC allocator and a switch allocator. The router demonstrated here uses the input buffering scheme and each input buffer contains M VCs. In synchronous VC routers, the $MP \times P$ crossbar is always simplified into a $P \times P$ crossbar and MUXes are added in each input buffer to select one VC per cycle. In asynchronous VC routers, VCs in the sample input buffer operate asynchronously and request different output ports concurrently. Using the $P \times P$ crossbar structure demands complicated switch al-

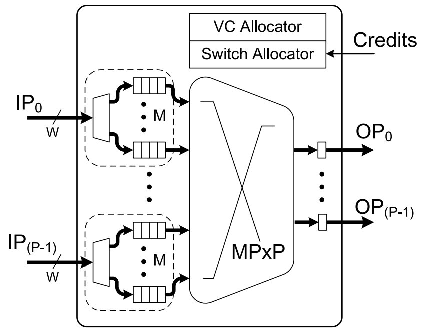


Fig. 9: A VC router with P ports of width W bits and M VCs in each port

locating algorithms [5] and leads to potential throughput waste.

Using the assumptions in Section III-A, the area model of a VC router is expressed as follows:

$$A_{IB,VC} = M \cdot A_{IB,WH} \tag{15}$$

$$A_{OB,VC} = A_{OB,WH} \tag{16}$$

$$A_{CB,VC} = (2MP^2 - P) \cdot (2W + 2) \cdot A_g \tag{17}$$

$$A_{A,VC} = (M^2P^2 + MP) \cdot A_{arb} \tag{18}$$

The area in Equation 18 includes two parts: the VC allocator and the switch allocator. As detailedly described in [17], the arbitration scheme of a fair VC allocator is $MP \times MP$ because MP output VCs are dynamically allocated to MP input VCs. The arbitration scheme of the switch allocator is $M \times P$ because every output port is competed by M VCs simultaneously.

Position of the critical cycle in the VC router depends on its internal architecture. If the output buffering scheme is in use, the critical cycle for data transmission is on the final stage of the output buffer, which includes the long wire between two adjacent routers. In that case, the maximal data throughput is affected by the global mapping. The output buffering scheme was used in MANGO and described in [7]. In this paper, we assume the VC router adopts the input buffering scheme shown in Fig. 9. Thus the VC router have the same critical cycle traversing the crossbar as the wormhole and the SDM routers. The latency of the critical cycle can be approximated as follows:

$$t_{C,VC} = t_{C,WH} \tag{19}$$

$$t_{CD,VC} = l_{CD} + l_C \cdot log_2(W/2) + k_{CD} \cdot MP \tag{20}$$

$$t_{AD,VC} = t_{AD,WH} \tag{21}$$

$$t_{CB,VC} = t_{CB,WH} \tag{22}$$

Using models and latencies provided in Section III-B, the estimated latencies for a 32-bit 5-port asynchronous VC router with four VCs are listed as follows (in unit of ns):

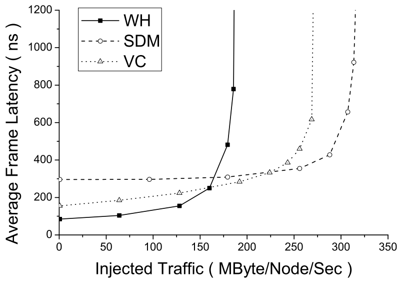
$t_C = 0.20$ $t_{CB} = 0.16$ $t_{CD} = 0.89$
$t_{AD} = 0.61$ $t_{CTL} = 0.78$
cycle period = 5.23
routing calculation = 0.44
VC allocator = 3.21
switch allocator = 0.78

As shown in the estimated latencies, asynchronous VC routers have the worst cycle period in all router architectures. The bandwidth of a VC is equal with that of an input buffer in the wormhole router. Both of them suffer from the worst ACK driver latency and the completion detection latency. The fact that the crossbar is reconfigured in every cycle introduces an extra switch allocation latency into every data cycle. The extra arbitration latency t_{CTL} comes from the switch allocation latency of the wormhole router assuming MUTEX-NET arbiters are in use. The VC allocator has the same arbitration scheme as the switch allocator in an SDM router; therefore, they have the same arbitration latencies.

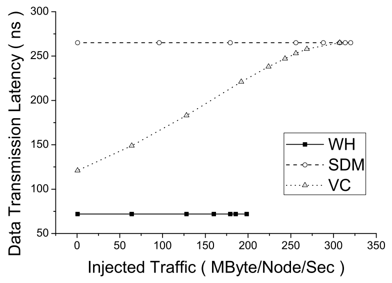
IV. Network Performance

We have built up latency accurate SystemC models for all router architectures using the latency models provided in the previous section. Fig. 10 shows the latency performance of all routers in an 8x8 mesh network. Since the best-effort performance is the only optimization target in this paper, all simulations use the random uniform traffic pattern. Every network node sends frames to other nodes uniformly in a poisson sequence. In the first test case, all routers are equipped with two stages of input buffers. A frame has a payload of 64 bytes. Four virtual circuits are implemented in each port in SDM routers and VC routers have four VCs per port.

As shown in Fig. 10a, both VC and SDM improve the throughput performance but SDM outperforms VC with the best throughput performance of 320 MByte/Node/Sec which is around 1.7 that of the wormhole router. However, SDM introduces extra frame latency when the network load is low because the available bandwidth of a virtual circuit is only a portion of the total bandwidth. The minimal frame latency of using SDM is 296 ns, which is 3.5 times that of using wormhole routers and 1.8 times that of using VC routers.



(a) Injected traffic vs. frame latency



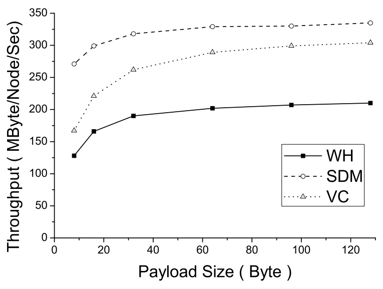
(b) Injected traffic vs. data latency

Fig. 10: Latency under various network load ($P = 5, W = 32, M = 4$)

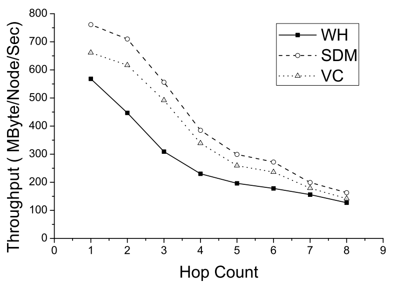
Frame latency comprises two parts: the latency for a head flit to reserve a path and the data transmission latency. Fig. 10b shows the data transmission latency. For wormhole and SDM, links and buffers are exclusively allocated to frames. Thus the data transmission latency is not affected by the network load. On the contrary, the data transmission latency of VC routers is tightly related to the network load because the physical link is shared by all VCs in a timing division manner. This result reveals the potential benefit of using SDM to reduce the latency jitter when latency guaranteed services are required.

Fig. 11a demonstrates the throughput performance of all router architectures with various payload lengths. The buffer depth in all routers is set to two. Since short frames introduce more control overhead, such as the target address in the head flit, the throughput performance rises with the payload length. This increasing is not linear. The throughput performance approaches its peak when the payload is larger than 64 bytes. Therefore, the payload size is fixed to 64 bytes in following simulations. SDM has outperformed VC. The SDM router provides the highest throughput performance of 335 MByte/Node/Sec (128 byte case).

Both VC and SDM alleviate the head-of-line (HOL) problem. Fig. 11b reveals the throughput performance with different communication distance. The throughput



(a)



(b)

Fig. 11: Throughput with various (a) payload length and (b) communication distance $(P = 5, W = 32, L = 2, M = 4)$

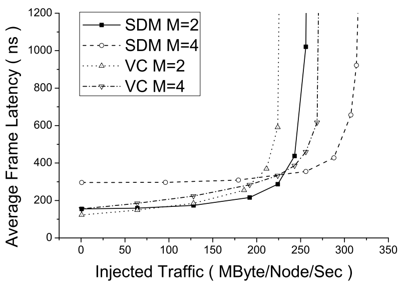


Fig. 14: Throughput with various number of virtual circuits or VCs $(P = 5, L = 2, W = 32)$

of all architectures drops significantly with the increasing communication distance. Both VC and SDM achieve better performance increment in local traffic patterns than in long distance communication patterns. When the communication distance is 8 hops, using VC shows little throughput boost but SDM still raises the throughput by 28%.

Increasing the buffer length in input buffers alleviates the HOL problem and raises throughput but introduces area overhead. Fig. 13 reveals the impact of increasing the buffer length. Both throughput and area overhead rise. Fig. 12c shows the gain of throughput per area unit. Higher gain indicates better area to throughput efficiency. The gain of all router architectures drops along with the increasing buffer length. It is not efficient to improve the throughput performance by adding buffers. The basic wormhole router shows the best area efficiency with short buffers. When long buffers are implemented $(L \geq 16)$, the SDM router has the best area efficiency. In all cases, VC routers are with the worst area efficiency.

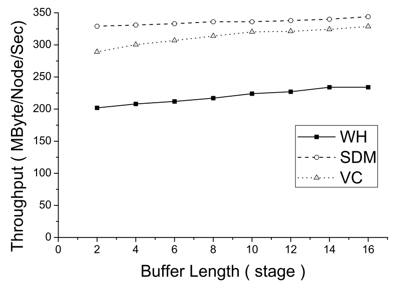
Besides adding buffers, increasing port bandwidth also raises throughput. However, the throughput improvement is not linear with bandwidth. Transmitting frames with the same payload length in wide pipelines suffers from the increased control overhead because the equivalent flit number per frame is decreased. Furthermore, increasing bandwidth introduces more synchronization overhead. Fig. 13 reveals the impact of increasing bandwidth. As expected, the area increases linearly with bandwidth but not throughput. Increasing bandwidth to more than 128 bits does not raise the throughput of wormhole router and VC routers. SDM routers still benefit from wider bandwidth because their C-element trees are shorter than those of wormhole and VC routers. Fig. 13c shows the gain of throughput per area unit. Although wormhole routers still show the best gain with narrow bandwidth, the gain of SDM routers increases with bandwidth until around 90 bits. In all cases, VC routers show the worst area efficiency.

The number of virtual circuits in an SDM router and the number of VCs in a VC router is an important design parameter. Increasing the number of virtual circuits or VCs allows more frames to be delivered concurrently and normally raises throughput. Fig. 14 demonstrates the latency versus injected traffic performance for SDM and VC routers with various number of virtual circuits and VCs.

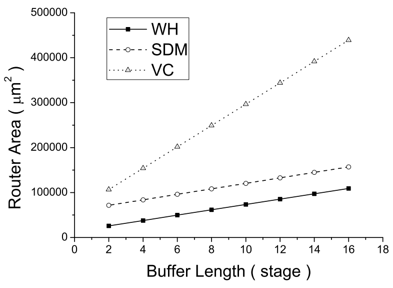
Both VC and SDM routers benefit from the increased number of VCs. Their throughput increases by 20.0% and 22.5% respectively. It is also shown in Fig. 14, the average frame latency of SDM routers in low network load increases with the number of virtual circuits. Since the total bandwidth of a port is fixed, increased the number of virtual circuits also reduces the effective bandwidth of a single virtual circuit. Data are serialized to fit the small bandwidth; therefore, the frame latency rises accordingly.

V. Conclusions

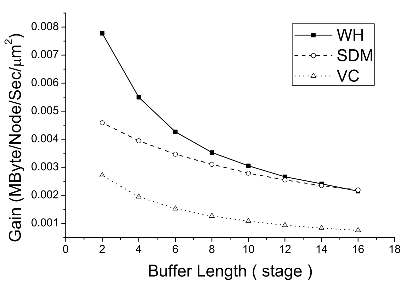
In this paper, we proposed and implemented an asynchronous spatial division multiplexing (SDM) router. The SDM router divides its buffers and data links into multiple virtual circuits. Every virtual circuit delivers a



(a)

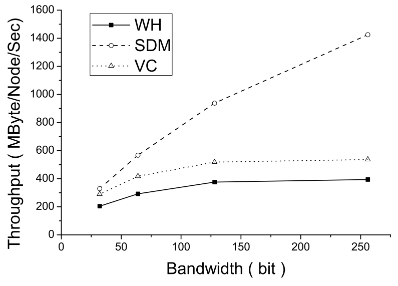


(b)

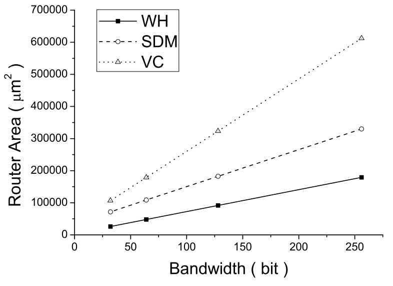


(c)

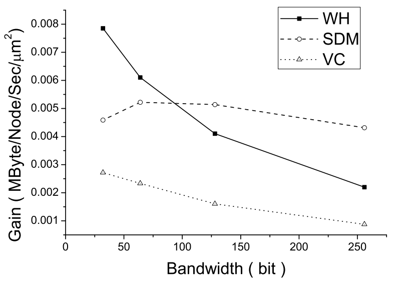
Fig. 12: (a) throughput, (b) router area and (c) gain with various buffer length ($P = 5, W = 32, M = 4$)



(a)



(b)



(c)

Fig. 13: (a) throughput, (b) router area and (c) gain with various bandwidth ($P = 5, L = 2, M = 4$)

frame using the basic wormhole flow control method. Since SDM reduces the effective bandwidth blocked by a pausing frame, it alleviates the HOL problem and, therefore, improves the throughput performance for best-effort traffic.

We have analysed the area overhead of using wormhole, SDM and VC in asynchronous routers and provided area models to estimate the router area with various configurations. We have also analysed the loop latency of the critical cycle in all router architectures and provided estimation models. Using practical parameters extracted from practical hardware implementations, several latency accurate SystemC models are built to exam the network performance of all router architectures in an 8x8 mesh network.

In all test cases, routers using the SDM flow control methods have outperformed routers using the VC control flow method. Throughput is related to payload size. Delivering data in long frames leads to higher throughput than short frames. Both adding buffers and increasing bandwidth raise throughput but increasing bandwidth shows better area to throughput efficiency than adding buffers. SDM introduces extra frame latency in low network load due to its serialized data transmission.

REFERENCES

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

[2] W. J. Dally and B. Towles, *Principles and Practices of interconnection networks*. San Francisco, CA: Morgan Kaufmann Publishers, 2004.

[3] 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.

[4] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, and D. Verkest, “Spatial division multiplexing: a novel approach for guaranteed throughput on nocs,” in *Proc of CODES+ISSS*, 2005.

[5] T. Felicijan and S. B. Furber, “An asynchronous on-chip network router with quality-of-service (QoS) support,” in *Proc. of IEEE International SOC Conference*, September 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*, 2005, pp. 54–63.

[7] 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.

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

[9] W. Song and D. Edwards, “A low latency wormhole router for asynchronous on-chip networks,” in *Proc. of ASP-DAC*, 2010, pp. 437–443.

[10] 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.

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

[12] M. B. Josephs and J. T. Yantchev, “CMOS design of the tree arbiter element,” *IEEE Transactions on VLSI*, vol. 4, December 1996.

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

[14] S. Golubcovs, D. Shang, F. Xia, A. Mokhov, and A. Yakovlev, “Modular approach to multi-resource arbiter design,” in *Proc. of ASYNC*, 2009, pp. 107–116.

[15] S. Golubcovs, D. Shang, and et al, “Multi-resource arbiter decomposition,” Newcastle University, Tech. Rep., 2009.

[16] D. Shang, F. Xia, S. Golubcovs, and A. Yakovlev, “The magic rule of tiles: virtual delay insensitivity,” in *Proc. of PATMOS*, 2009.

[17] L.-S. Peh and W. J. Dally, “A delay model and speculative architecture for pipelined routers,” in *Proc of HPCA*, January 2001, pp. 255–266.