Asynchronous Spatial Division Multiplexing Router

Wei Song and Doug Edwards

School of Computer Science, the University of Manchester, Oxford Road, Manchester M13 9PL 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. Performance comparisons have been made with different configurations of payload size, communication distance, buffer size, port bandwidth, network size and number of VCs/virtual circuits. Compared with VC, SDM achieves higher throughput with lower area overhead.

Keywords: Networks-on-chip, Asynchronous circuits, Spatial division multiplexing, Virtual channel

1. Introduction

Network-on-Chip (NoC) [1] is the state-of-the-art communication fabric for current and future multiprocessor SoC (MPSoC) systems which contain tens to hundreds of cores. The on-chip network can be a synchronous network where cores and routers are driven by the same or several global clocks, or an asynchronous network where self-timed routers seamlessly link cores running at different frequencies. Thanks to the timing assumptions allowed by the global clock and mature EDA tools, current synchronous NoCs are fast and area efficient. However, the clock tree is power consuming and this gets worse as technology feature size continues to shrink. Coping with the process variation and unreliable cell latencies of future CMOS technologies is another problem. Timing closure is already challenging for traditional static timing analysis techniques. Asynchronous circuits, especially the quasi-delay-insensitive (QDI) circuits [2], are naturally tolerant to process variation and less power consuming than their synchronous counterpart. When synchronous cores are connected by a global asynchronous on-chip network, the network interfaces of cores are unified automatically and the overall design time is shortened [3].

Flow control is one of the major NoC research issues. Most router designs favour 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 a VC

is allocated to each frame. 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 for the following reasons:

(1) Every VC is a separated set of buffers. VC routers normally need deep buffers to compensate the credit loop latency [4] and maximize throughput. Most QDI routers use 4-phase 1-of-4 or dual-rail handshake protocols [5, 6, 7, 8]; therefore, every 1-bit full buffer is implemented using four C-elements and proper completion detection gates, which consume more space than a flip-flop. Each extra VC consumes more area in asynchronous routers than in synchronous ones.

(2) To form a wide buffer stage, QDI routers synchronize multiple 1-of-4 or dual-rail buffers using a C-element tree in every buffer stage [9, 10, 5, 6, 7]. This C-element tree increases the latency of each buffer stage and compromises speed. In a wormhole router where the internal crossbar is reconfigured per frame, the channel slicing technique [8] is capable of removing the C-element tree and reducing latency. However, the virtual channel flow control method forces the internal crossbar to be reconfigured per flit. This prevents the use of channel slicing. Furthermore, the latency of reconfiguring the crossbar is now added into the transmission time of each flit rather than only the head flit in a wormhole router.

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 [11, 12]. 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 achieves better throughput with less area overhead 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, the channel slicing technique can be adopted in SDM routers to boost throughput because the crossbar is still reconfigured once per frame for each virtual circuit as the wormhole router does.

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 2 reviews previously published network and router designs. Section 3 demonstrates the hardware architecture of the proposed asynchronous SDM router and the channel slicing technique. Section 4 provides area and latency estimation models and verifies them

with the results from practical implementations. Based on these models, Section 5 reveals the network performance of using various flow control methods with different configurations. Finally the paper is concluded in Section 6.

2. Previous Work

An on-chip network can be implemented synchronously or asynchronously. Although most of current NoCs are synchronous, asynchronous NoCs do have some distinctive advantages. When a NoC contains multiple clock domains, a synchronous NoC suffers from the extra speed penalty of multiple synchronization latencies on its data links. Asynchronous NoCs deliver data according to handshake protocols. They introduce no synchronization issues until the boundary in asynchronous/synchronous interfaces. Furthermore, the handshake scheme makes asynchronous circuits naturally tolerant to process variation and they dissipate zero dynamic power in idle status. Thanks to these benefits, ITRS [13] predicts that 40% of the global signalling in SoC platforms will be performed asynchronously by the year 2020. Our research is mostly concentrating on asynchronous NoCs.

An asynchronous NoC is a globally asynchronous and locally synchronous (GALS) system where synchronous cores are connected through an asynchronous on-chip network [3, 6]. Network interfaces in such networks handle safe and reliable data transfer between synchronous and asynchronous domains while serving as a buffer and a protocol translator between cores and routers. One approach of implementing a safe interface is to drive each individual synchronous core by a pausible-clock generator inside its network interface [14, 15]. The clock generator ensures that no clock pulse is generated during data transmission, which avoids metastability. However the clock generator may degrade the throughput performance when the local clock tree is deep [15]. An alternative approach to achieve high throughput interface is using asynchronous FIFOs [14, 16] but they introduce extra latency. In either approach, the communications between network interfaces and routers are asynchronous data transmissions that comply with certain handshake protocols.

Virtual channel and SDM are not the only flow control methods in NoC research. The time division multiplexing (TDM) flow control method has been frequently utilized in NoCs providing communications with guaranteed bandwidth and predictable latency, such as the Philips Æthereal NoC [17]. TDM divides time into slots and allocates these time slots to different communications. Since the allocation is foreknown and the maximal waiting time for each slot is predictable, communication deadline is guaranteed. Asynchronous circuits are insensitive to latency. The time slot is hard to define asynchronously and TDM is unavailable to asynchronous NoCs. Besides TDM, circuit switched networks [18] also guarantee the communication latency. Every communication in a circuit switched network needs to reserve a path before any data transmission but this reservation process is prone to waste throughput. It is not favourable to best-effort traffic. The Rotary router [19] used the virtual cut-through flow control method and demonstrated efficient best-effort

performance for chip multi-processor (CMP) interconnection networks. However, the virtual cut-through method requires an input buffer to be large enough for a single frame. Unlike CMP systems, frames in SoC systems can be extremely long incurring area overhead.

Undoubtedly, virtual channel [20] is one of the most successful flow control methods. Most asynchronous routers utilize virtual channels. In the QoS router [5] and ANoC [6], qualities of services (QoS) were supported by assigning VCs with different priorities. The VC with the lowest priority is reserved for best-effort traffic. Communications with higher priorities reserve other VCs and go through routers with lower latency than best-effort traffic. The MANGO router [21] employing the Asynchronous Latency Guarantee (ALG) algorithm ensured a worst case latency for each priority level. The QNoC router [22] extended its QoS support by implementing multiple VCs in each priority level. However, none of these VC routers tried to expand the throughput of best-effort traffic and no publication has so far compared the throughput increment of using virtual channels in asynchronous routers against the basic wormhole method. Other than VC routers, an asynchronous wormhole router with customized FIFOs was implemented in [7]. The Chain interconnection network [9] was employed in SpiNNaker CMP chips [10] to connect all on-chip processors to a central router.

Up to now, SDM has not been implemented asynchronously yet. The first synchronous SDM router was implemented in [11]. It utilizes SDM to support latency guaranteed communications. Since every virtual circuit in an SDM router exclusively occupies a portion of the total bandwidth, communication latency is predictable as in TDM and circuit switched networks. For best effort traffic, the investigation in [23] revealed that SDM highly increases the throughput performance at the expense of a small latency overhead.

3. Implementation

3.1. Overall Architecture

An asynchronous SDM router is illustrated in Figure 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, 6, 21, 22, 8].

The 4-phase 1-of-4 protocol is used in our router implementations. Many handshake protocols have been used to build asynchronous routers. The 4-phase bundled-data protocol has been used in MANGO [21], QNoC [22] and ASPIN [7]. The 4-phase dual-rail protocol has been used in ASPIN [7]. The 4-phase 1-of-4 protocol has been used in CHAIN [9], QoS [5] and ANoC [6]. Bundled-data protocols only work under

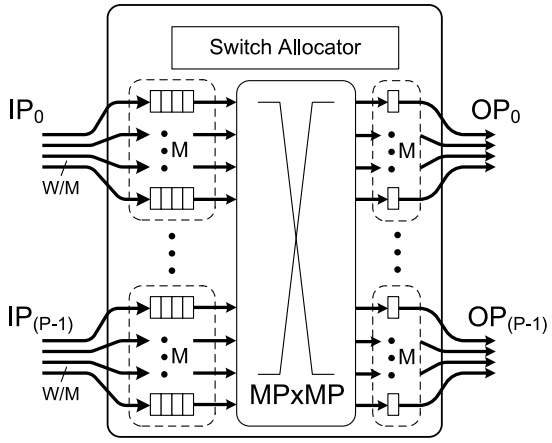


Figure 1: An SDM router with P ports of width W bits and M virtual circuits in each port

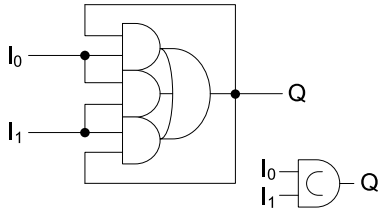


Figure 2: The implementation and symbol of a C-element

cautious timing constraints and the matched delay lines are vulnerable to process variation. Compared with the dual-rail protocol, the 1-of-4 protocol is more efficient in both power and area consumption.

One of the primitives used in QDI circuits is the C-element. The implementation of a 2-input symmetric C-element using standard cells is shown in Figure 2. It behaves like an AND gate with memory. Its output is set to 1/0 when all its inputs are 1/0. During intermediate states when inputs have different values, the output retains the previous value. C-elements are used as latches to store data or as synchronizers to synchronize parallel transitions.

A wide QDI buffer stage is normally formed by synchronizing multiple bit-level buffer stages. Figure 3 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.

3.2. Input buffers

Every input virtual circuit is a fully functional input buffer as in a wormhole router. As shown in Figure 4, 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

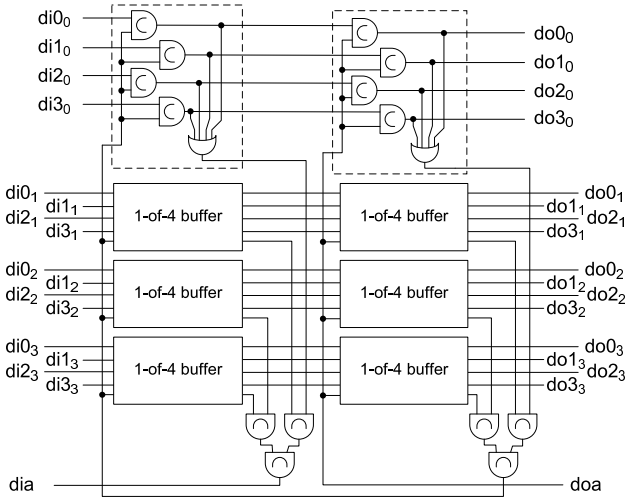


Figure 3: Two 8-bit QDI buffer stages

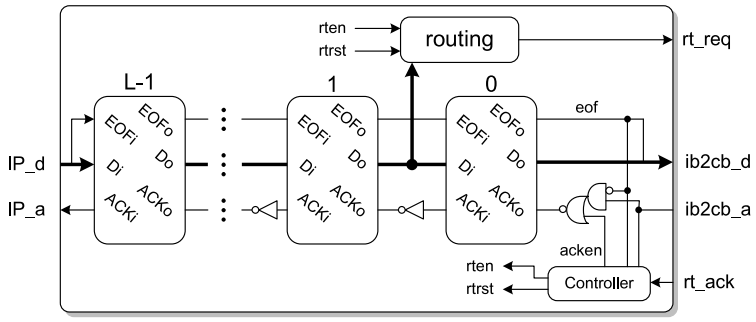
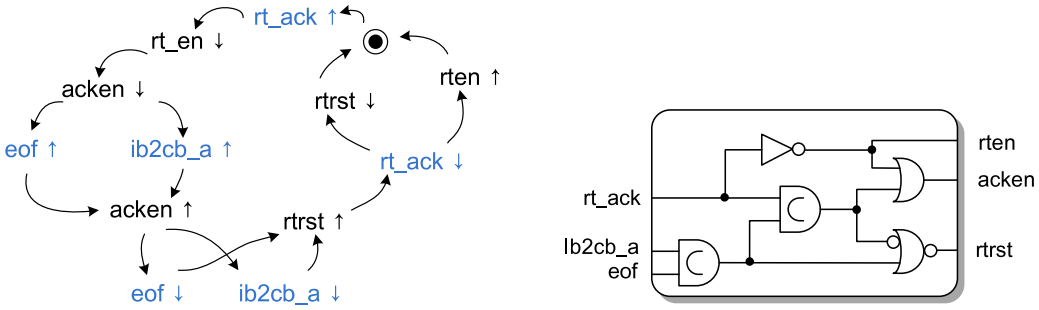


Figure 4: Structure of an input buffer

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. Figure 5 illustrates the signal transition graph (STG) of the buffer controller and the circuit generated using Petrify [24].



(a) STG (b) Schematic

Figure 5: Buffer controller

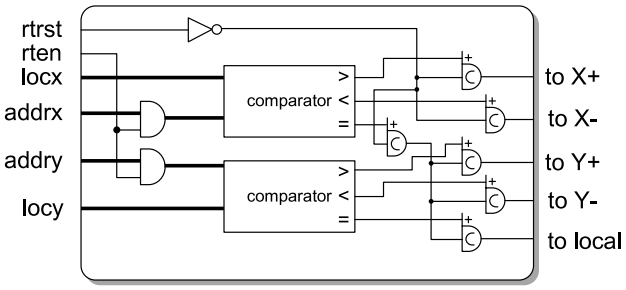


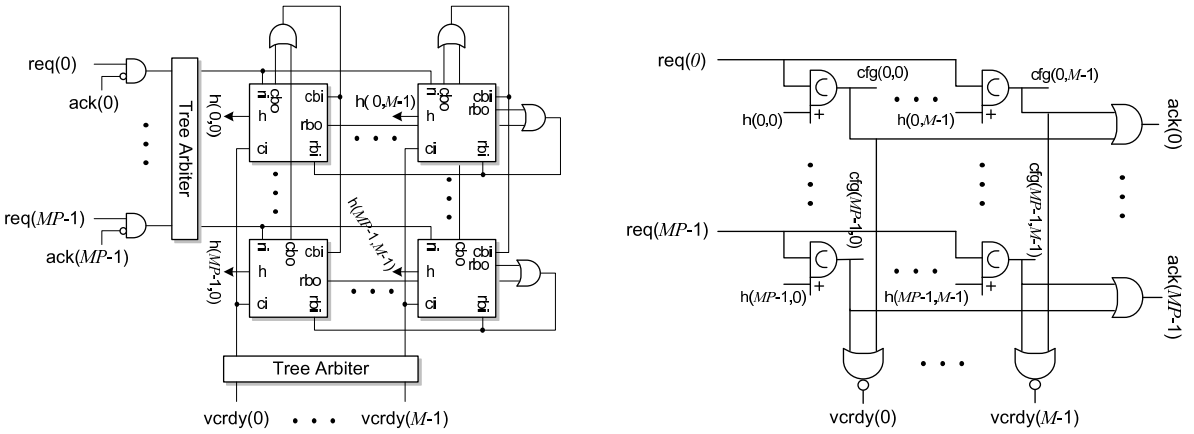
Figure 6: Routing calculation circuit

Figure 6 shows a routing calculation circuit using the deterministic XY routing algorithm. 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.

SDM is applicable to all types of routing algorithms. Deadlock-free algorithms are easy to support. Adaptive algorithms lead to deadlock issues. Similar to VC routers, SDM routers can reserve a virtual circuit on every link as an escaping channel or label virtual circuits with different resource classes to avoid deadlocks [4]. These deadlock avoidance techniques require unfair arbitration schemes and lead to traffic imbalance. As this paper concentrates more on the throughput benefit due to using different flow control methods rather than on routing algorithms, all the router architectures described here use the deterministic XY routing algorithm.

3.3. Switch allocator

Although there is only one switch allocator shown in Figure 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 $MP \times M$.



(a) Allocation matrix

(b) Configuration capture matrix

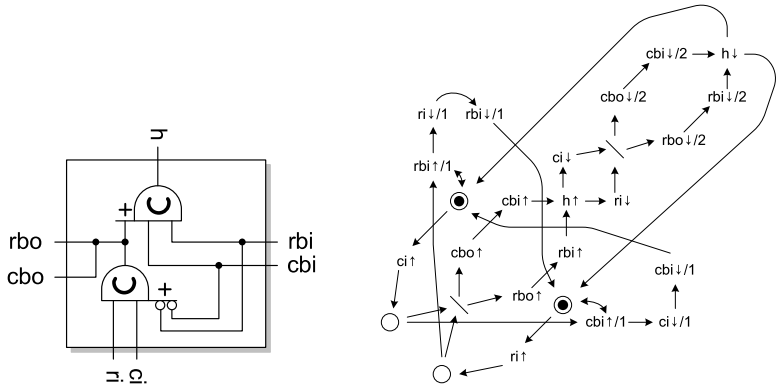
Figure 7: Switch allocator

Traditional asynchronous arbiters, such as the MUTEX-NET arbiter [25, 22], the tree arbiter [26] and the ring arbiter [27], are capable of allocating only one resource to multiple clients. The only two allocators that allocate multiple resources to multiple clients are the virtual channel admission controller (VAVC) presented in QNoC [22] and the multi-resource arbiter [28, 29, 30]. The VAVC allocator treats all clients fairly but resources are selected by an unbalanced static priority arbiter (SPA) [31]. The multi-resource arbiter is the only allocator that can fairly allocate multiple resources to multiple clients.

Figure 7 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 Figure 7(a) is a tile-based multi-resource arbiter [29]. 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 one at 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 [26] is added on both rows and columns to guarantee the one-hot condition in both of them.

Figure 7(b) 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



(a) Schematic (b) STG

Figure 8: Arbitration tile

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 released when the request req[r] is withdrawn by the input virtual circuit.

The internal structure of a tile in the allocation matrix and its STG are shown in Figure 8 [29]. ri and ci are the row request and column request respectively. rbi and cbi are the blocking signals indicating a match is found in the corresponding row or column. They are driven by the OR gate trees shown in Figure 7(a). When a new match is found, the rbo and cbo signals in the corresponding tile is driven to block all other tiles in the same row and column. When the tile receive a positive pulse on both rbi and cbi, other tiles have been firmly blocked and the match result h is safely triggered.

The use of such a tile matrix may seem over-complicated. However, asynchronous circuits work without any timing assumptions. The tree arbiters on row and column run in totally unsynchronized sequences. Without the tile matrix, the tree arbiter on row may withdraw the row request very slowly leading to a false match with the new winner of the tree arbiter on column. Similar faults occur when the tree arbiter on column is too slow. The detailed verification and optimization of the multi-resource arbiter is described in [28, 29, 30].

3.4. 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 by AND gates and OR gates, such as the 4×3 crossbar shown in Figure 9. The standard MUX and DEMUX gates are not available for asynchronous circuit due to their possible glitches.

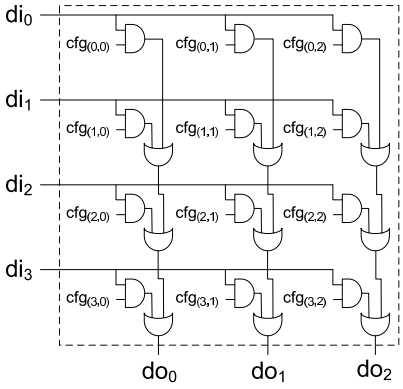


Figure 9: A 4x3 crossbar with 1-bit bandwidth

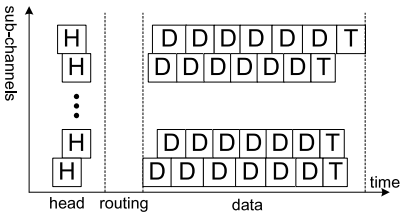


Figure 10: Data flow in a sliced buffer

3.5. Channel slicing

First presented in [8], channel slicing is a technique capable of removing the C-element tree in every buffer stage as long as the basic wormhole flow control method is used. It separates synchronized wide buffer stages into parallel bit-level buffer stages. A bit-level pipeline is namely a sub-channel.

Although channel slicing allows sub-channels running in parallel, they are still synchronized once per frame. As shown in Figure 10, once the synchronization among sub-channels is eliminated, the data bits being delivered to the same buffer stage (bit-level buffers in the same column) may belong to different flits. This does not incur any control problem unless the head flit is affected. Therefore, extra control logic is added to guarantee the correct head flit is received by re-synchronizing all sub-channels. When a path is successfully reserved in the crossbar, sub-channels run in parallel again until the next head flit.

As shown in Figure 11(a), an input buffer is divided into N sub-channels. Each of them is an independent pipeline with its own sub-channel controller and its ACK line. A sub-channel controller has similar function to the buffer controller in Figure 5. Since sub-channels are required to be re-synchronized when a new head flit is arriving, an EOF bit is added on each sub-channel. Thus sub-channel controllers can identify the tail flit. Data flits are vertically separated and transmitted by all sub-channels concurrently. The routing calculation circuit still reads the head flit from the 1st stage but now the 1st stage is separated into N sub-channel stages. The routing control signals rten and rtrst are driven by the extra routing controller.

Connections between the routing controller and sub-channel controllers are shown in Figure 11(b). Unlike

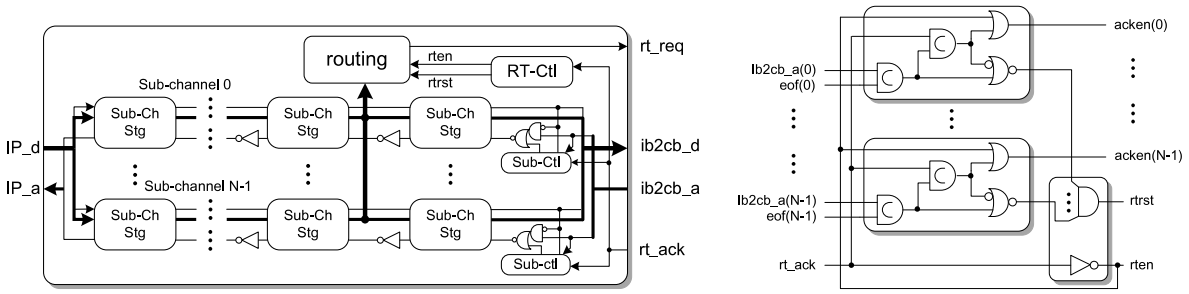


Figure 11: (a) a sliced input buffer and (b) its controllers

the buffer controller, the sub-channel controller reads rten as an input as it is now generated by the routing controller. Sub-channels still produce their own rtrst signals once their own share of the tail flit is transmitted. The actual rtrst signal is driven by an AND gate tree inside the routing controller to ensure that the reserved path is only released when all parts of the tail flit is transmitted. The AND gate tree should be substituted by a C-element tree if the strict QDI condition is required. Since the transmission time of a frame is far longer than the propagation time of a C-element tree, it is simplified to reduce control latency.

Similar to input buffers, output buffers are divided into sub-channels. No extra control logic is needed as the case for a wormhole router. The channel slicing technique improves throughput with extra area overhead. When the width of a single sub-channel is 2 bits, the C-element tree in the completion detection circuit is completely removed. However, an extra sub-channel controller, an EOF bit and an ACK line are added for each sub-channel. These area overhead will be analysed in the next section.

4. Hardware Performance

In this section, a basic wormhole router, an SDM router using synchronized buffers and an SDM router using channel slicing are implemented using the Faraday standard cell library based on the UMC 0.13 μm technology and standard synthesis tools. Some well-recognized configurations are adopted in our router implementations. 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. The 8-bit address is enough to identify any node in a 16x16 network using the XY routing algorithm. 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.

It is possible to support more than four virtual circuits with area and latency overhead. For instance, a routing algorithm needs to analyse N bits for one routing decision. If the bandwidth of each virtual circuit is less than N bits, these N bits are delivered by multiple flits and the routing calculation circuit waits for

all these flits before making a routing decision. This prolongs the control latency and the minimal buffer depth. Nevertheless, it is possible to support all types of routing algorithms with possible overhead.

4.1. 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:

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

where A_C, A_{EOF},A_{RC} and A_{BC} 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:

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

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

If the channel slicing technique is used in an SDM router (SDMCS), a buffer stage is further separated into sub-channels. Every sub-channel has its own sub-channel controller which is approximately the same area as a buffer controller. The area of the routing controller is negligible. Thus the area calculation is:

$$A_{IB,SDMCS} = M \cdot [\frac{WL}{2M} \cdot (5A_C + A_{EOF}) + \frac{W}{2M} \cdot A_{CTL} + A_{RC}] \tag{6}$$

$$A_{OB,SDMCS} = 2.5WA_C + 0.5WA_{EOF} \tag{7}$$

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 Figure 9, the area of the crossbars in all routers is as follows:

$$A_{CB,WH} = (2W + 2)(2P^2 - P)A_g \tag{8}$$

$$A_{CB,SDM} = (\frac{2W}{M} + 2)(2M^2P^2 - MP)A_g \tag{9}$$

$$A_{CB,SDMCS} = \frac{3W}{M}(2M^2P^2 - MP)A_g \tag{10}$$

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 for all allocators. Therefore, the area is roughly proportional to the arbitration scheme. The area of the switch allocators in all routers can be estimated as:

$$A_{A,WH} = P^2A_{arb} \tag{11}$$

$$A_{A,SDM} = A_{A,SDMCS} = M^2P^2A_{arb} \tag{12}$$

where A_{arb} is the equivalent area overhead of a single arbitration point in the arbitration scheme. Note that different arbiter structures should use different area estimation models to reach an accurate estimation.

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

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

The practical area consumption and the estimation error rates of the proposed models are illustrated in Table 1. In practical implementations, the crossbar and the switch allocator are simplified. All disabled turn models in the XY routing algorithm are removed; the actual estimation processes have been adjusted accordingly. 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 and the multi-way MUTEX arbiter [25] is utilized to reduce arbitration latency and area overhead. As shown in Table 1, 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. Figure 12 shows the area estimation error rates of routers with various configurations. In all configurations, the estimation error rate is lower than 4.9%. Therefore our model is adequate to estimate the area consumption for routers with various configurations.

There are some limitations for this area model. It is well known that technology affects area consumption significantly. Even two cell libraries with the same technology scale but from different foundries behave

Table 1: Area consumption (μm^2)

	WH	err(%)	SDM	err(%)	SDMCS	err(%)
Input Buffers	14,303	0.0	21,995	-0.4	25,953	-0.1
Output Buffers	5,935	0.0	6,000	1.7	6,540	3.4
Crossbar	4,356	0.0	21,744	-0.2	28,992	-0.2
Switch Allocator	772	78.2	22,208	-0.9	22,122	-0.5
Total	25,366	2.4	71,956	-0.3	83,615	0.0

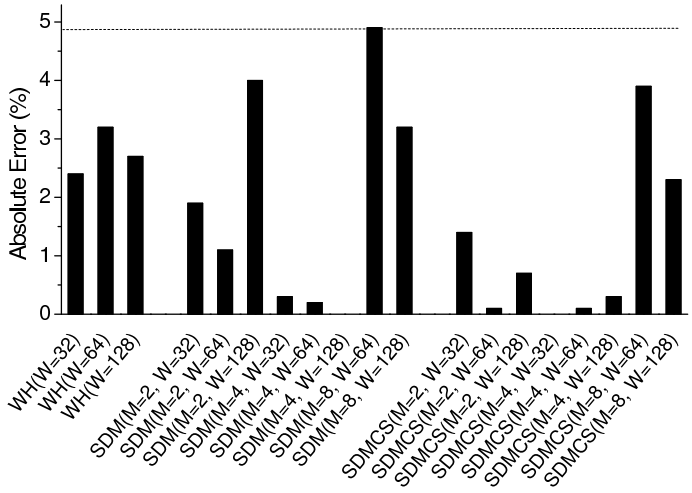


Figure 12: Area estimation error with various configurations

slightly differently. To achieve an accurate area estimation, all parameters must be extracted again when a different cell library is in use. The area model for arbiters is not accurate and it is the major source of area estimation error. As shown in Figure 12, wormhole routers and SDM routers with two or eight virtual circuits have higher error rates than SDM routers with four virtual circuits because they have a different arbiter structure or arbitration scheme. Fortunately, we are more interested to the overall area of a router instead of its area break down. Normally arbiters consume a small portion of the total area. This inaccurate arbiter model is enough for the performance analyses in this paper.

SDM introduces area overhead over the wormhole router. If $W \gg M$ and $2MP \gg 1$, the area of the crossbar can be simplified from Equation 8 and Equation 9:

$$A_{CB,SDM} \approx M \cdot A_{CB,WH} \tag{13}$$

The area of the crossbar is proportional to the number of virtual circuits M. SDM also introduce $M-1$ routing calculation circuit and buffer controllers. The area of the switch allocator also increases to M^2 times of that in a wormhole router, indicated by Equation 12. In practical implementations, the area of the SDM router is 2.84 times that of the wormhole router. Using channel slicing in an SDM router introduces more

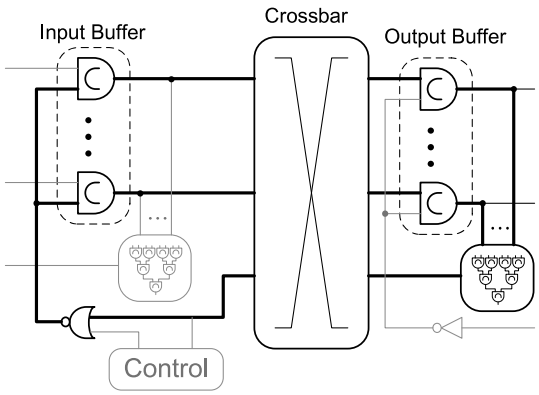


Figure 13: The critical cycle in asynchronous routers

controllers in input buffers and enlarges the wire count. As shown in Table 1, the area of input buffers, output buffers and the crossbar is increased by 18%, 9% and 33% respectively. Although SDM and channel slicing introduce area overhead, they improve throughput performance significantly. Detailed simulation results are shown in Section 5.

4.2. Latency analysis

Critical cycle: for an asynchronous pipeline, the 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 obviously 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 Figure 13 where the loop path is highlighted in bold lines. To transmit one data flit on a 4-phase pipeline, a series of transitions 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:

$$T = 4t_C + 4t_{CB} + 2t_{CD} + 2t_{AD} + t_{CTL} \tag{14}$$

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 Figure 13) and t_{CTL} is the possible latency caused by the controller.

To ease the latency estimation, 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.

$$t_C = \begin{cases} l_C + k_C(P+1) & \text{wormhole,} \\ l_C + k_C(MP+1) & \text{SDM and SDMCS.} \end{cases} \tag{15}$$

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.

$$t_{AD} = \begin{cases} l_{AD} + k_{AD}(2W+1) & \text{wormhole,} \\ l_{AD} + k_{AD}(2W/M+1) & \text{SDM,} \\ l_{AD} + k_{AD} \cdot 5 & \text{SDMCS.} \end{cases} \tag{16}$$

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.

$$t_{CB} = \begin{cases} l_{CB} + k_{CB} \cdot log_2(P) & \text{wormhole,} \\ l_{CB} + k_{CB} \cdot log_2(MP) & \text{SDM and SDMCS.} \end{cases} \tag{17}$$

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.

$$t_{CD} = \begin{cases} l_{CD} + l_C \cdot log_2(W/2) + k_{CD} \cdot P & \text{wormhole,} \\ l_{CD} + l_C \cdot log_2(\dfrac{W}{2M}) + k_{CD} \cdot MP & \text{SDM,} \\ l_{CD} + k_{CD} \cdot MP & \text{SDMCS.} \end{cases} \tag{18}$$

where l_{CD} and k_{CD} are 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, a buffer controller or a sub-channel controller only halts a data path once per frame. Data paths are running with full speed during data transmission; therefore, the extra control latency for these routers is zero. However, the controller in a VC router does halt the data path in every cycle to reconfigure the crossbar.

We have extracted all parameters through the post-synthesis simulations. The practical implementation has removed all unnecessary turns in the crossbar according to the XY routing algorithm. The actual estimation models are adjusted to the practical gate fanout. Wire latency is counted into gate latency automatically using wire load models. The extracted parameters are listed as follows (in unit of ns):

Table 2: Speed performance (ns)

	WH	err	SDM	err(%)	SDMCS	err(%)
cycle period	4.22	-2.1	4.10	-3.0	3.12	4.4
router latency	2.29		2.49		2.66	
routing calculation	0.44		0.51		0.50	
switch allocation	0.78		3.21		3.28	
t_C	0.22	-9.1	0.34	-5.9	0.29	10.3
t_{CB}	0.16	1.3	0.26	-3.8	0.24	4.2
t_{CD}	0.79	7.6	0.57	4.2	0.30	-2.0
t_{AD}	0.53	-6.6	0.27	-5.5	0.18	8.3
t_{CTL}	0.00		0.00		0.00	

$$l_C = 0.15 \quad k_C = 0.01 \quad l_{CB} = 0.074 \quad k_{CB} = 0.044$$
$$l_{CD} = 0.23 \quad k_{CD} = 0.004 \quad l_{AD} = 0.17 \quad k_{AD} = 0.005$$

Table 2 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 2 also shows the estimation error rates. The latency estimation has higher 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. To ease the area and latency estimation, no mapping and driving strength altering optimization is enabled in the synthesis process, and all C-elements and MUTEXes are manually mapped. With proper optimization, the speed performance of all routers can be improved. Specifically, the cycle period of the same wormhole router has been optimized to 2.9 ns in [8]. Figure 14 shows the speed estimation error with various configurations. Most of the errors are lower than 6.5%. All the configurations with significant error rate use wide data pipelines (bandwidth $>$ 32 bits). These wide pipelines cause design rule violations (max capacitance) on all ACK lines during the standard synthesis procedure. Thus buffer trees are inserted on ACK lines to eliminate these violations. However, buffer trees do not fit the linear latency model of t_{AD} and lead to a significant estimation error. To cope with this error, all simulations in Section 5 that use bandwidth larger than 32 bits are carefully compensated. Similar to the area estimation model, technology and cell library have strong impact on the accuracy and all parameters should be extracted again when a different cell library is in use.

As shown in Table 2, channel slicing significantly reduces the cycle period by 24%. It removes all C-elements in the completion detection circuit and it reduces the fanout of every ACK line. As shown in the practical latencies and the latency estimation models, t_{CD} and t_{AD} are reduced accordingly.

The speed performance comparison between SDM and wormhole routers is more interesting. 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

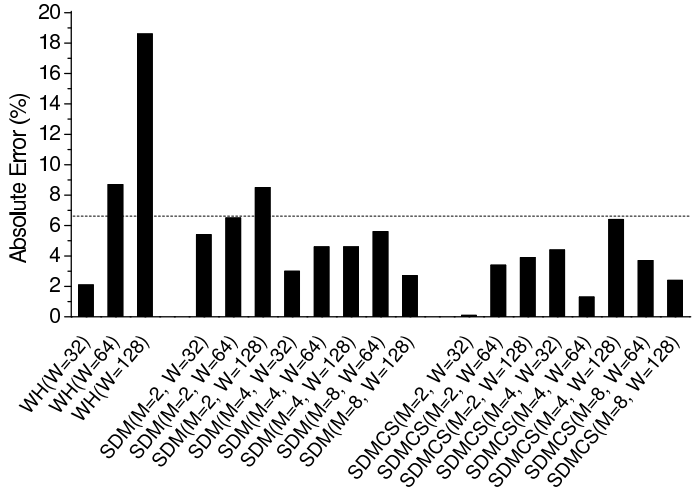


Figure 14: Speed estimation error with various configurations

increases the port number of the crossbar, which leads to longer C-element propagation delay, increases the crossbar traversing latency and raises 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 than the wormhole router. The latency reduction has compensated the extra latency introduced by the crossbar in our implementation.

5. Network Performance

In this section, area and latency models of VC routers are inferred from the router architecture. Using these models, the throughput and latency performance of all router architectures are analysed in an 8x8 mesh network.

5.1. VC router

As depicted in Figure 15, 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. If the output buffering scheme is in use, such as the MANGO router, each output buffer contains M VCs and the connection model of the crossbar is $P \times MP$ [21]. 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 same input buffer operate asynchronously and request different output ports concurrently. Using the $P \times P$ crossbar structure demands complicated switch allocating algorithms [5] and leads to potential throughput waste.

Using the assumptions in Section 4.1, the area model of a VC router is expressed as follows:

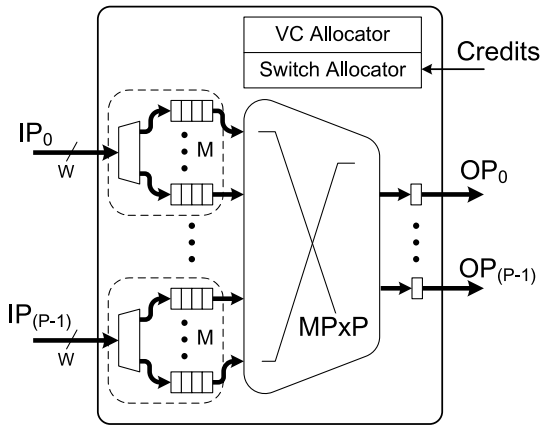


Figure 15: A VC router with P ports of width W bits and M VCs in each port

$$A_{IB,VC} = M \cdot A_{IB,WH} \tag{19}$$

$$A_{OB,VC} = A_{OB,WH} \tag{20}$$

$$A_{CB,VC} = (2MP^2 - P) \cdot (2W + 2) \cdot A_g \tag{21}$$

$$A_{A,VC} = (M^2P^2 + MP) \cdot A_{arb} \tag{22}$$

The area in Equation 22 includes two parts: the VC allocator and the switch allocator. As described in [32], 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 an output port is requested by maximal M input VCs simultaneously.

The 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 [21].

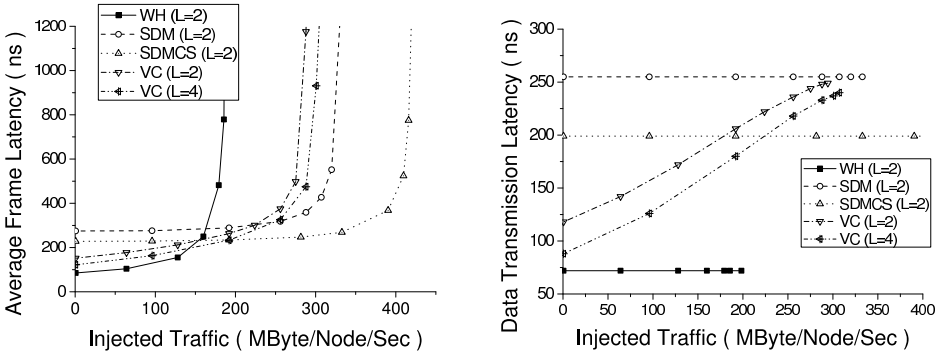
In this paper, we assume VC routers adopt the input buffering scheme shown in Figure 15. Thus VC routers have the same critical cycle traversing the crossbar as wormhole and the SDM routers. The latency of the critical cycle can be approximated as follows:

$$t_{C,VC} = t_{C,WH} \tag{23}$$

$$t_{CD,VC} = l_{CD} + l_C \cdot log_2(W/2) + k_{CD} \cdot MP \tag{24}$$

$$t_{AD,VC} = t_{AD,WH} \tag{25}$$

$$t_{CB,VC} = t_{CB,WH} \tag{26}$$



(a) Injected traffic vs. frame latency

(b) Injected traffic vs. data latency

Figure 16: Latency under various network load ($P = 5, W = 32, M = 4$)

Using models and latencies provided in Section 4.2, the estimated latencies for a 32-bit 5-port asynchronous VC router with four VCs are listed as follows (in unit of ns):

$t_C = 0.20$ $t_{CB} = 0.16$ $t_{CD} = 0.89$ $t_{AD} = 0.50$ $t_{CTL} = 0.78$

cycle period = 5.01 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 of 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 increased crossbar introduces extra latency as an SDM router does. The fact that the crossbar is reconfigured in every cycle obstructs the use of channel slicing and introduces extra switch allocation latency in 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 should have similar arbitration latencies.

5.2. Performance analysis

We have built up latency accurate SystemC models for all router architectures using the latency models provided in previous sections. Figure 16 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 excepted the VC router which is also simulated with four stages. 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 Figure 16(a), both VC and SDM improve the throughput performance but SDM outperforms VC. The SDM router using channel slicing has the best throughput performance of 436 MByte/Node/Sec

which is around 2.1 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 275 ns, which is 3.2 times that of using wormhole routers and 1.8 times that of using VC routers. Channel slicing reduces this latency to 228 ns.

VC routers suffer from the credit loop latency when their input buffers are too short. Note that every asynchronous buffer stage is a half buffer stage (latch). The effective full buffer depth for a two stages input buffer is one. Asynchronous VC routers normally use the credit based backpressure method [5, 21, 6, 22]. Inferred from the synchronous credit loop calculation in [4], the credit loop latency t_{crt} in an asynchronous VC router can be expressed as:

$$t_{crt} = Lt_C + 2t_w + t_{CTL} + T \tag{27}$$

where t_w is the latency of the long wire between two routers. Adopting the long wire latency in [7], the credit loop latency for our VC router implementation is around 1.3 cycle periods. As a result, at least two full buffer stages are required to avoid the credit stall under low network load. Figure 16(a) shows that using four stages of input buffers (two stages of full buffer), the frame latency is reduced to 122 ns; however, it does not raise throughput significantly and introduces extra area overhead by doubling the buffer size.

Frame latency comprises two parts: the latency for a head flit to reserve a path and the data transmission latency. Figure 16(b) 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 network load. On the contrary, the data transmission latency of VC routers is tightly related to network load because a physical link is shared by all VCs in a timing division manner. This result reveals the potentiality of using SDM to reduce the latency jitter when latency guaranteed services are required.

Figure 17 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, throughput rises with payload length. This increase is not linear. The throughput performance approaches its peak when payload is larger than 64 bytes. Therefore, the payload size is fixed to 64 bytes in all the following simulations. SDM outperforms VC. The SDM router using channel slicing provides the highest throughput performance of 442 MByte/Node/Sec (128 byte case).

Both VC and SDM alleviate the head-of-line (HOL) problem. Figure 18 reveals the throughput performance with different communication distance. In this simulation, network nodes send frame with fixed size frames (64 bytes) uniformly to all nodes certain hops away. The throughput of all architectures drops significantly with the increasing communication distance as longer distance introduce more contention. 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 33.9%. Channel slicing further lifts the throughput by 36.1%.

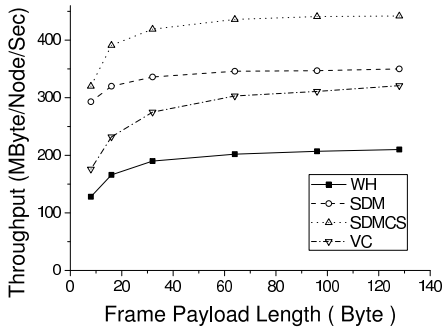


Figure 17: Throughput with various payload length ($P = 5, W = 32, L = 2, M = 4$)

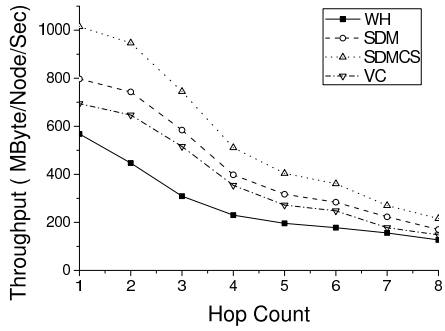
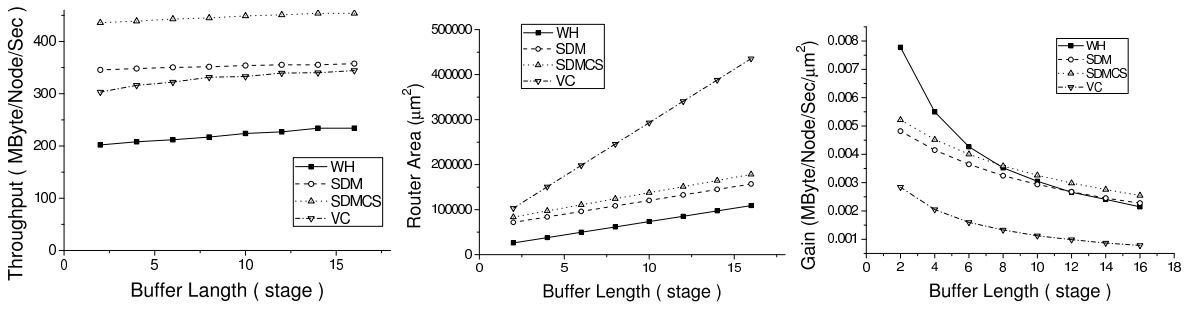


Figure 18: Throughput with various communication distance ($P = 5, W = 32, L = 2, M = 4$)

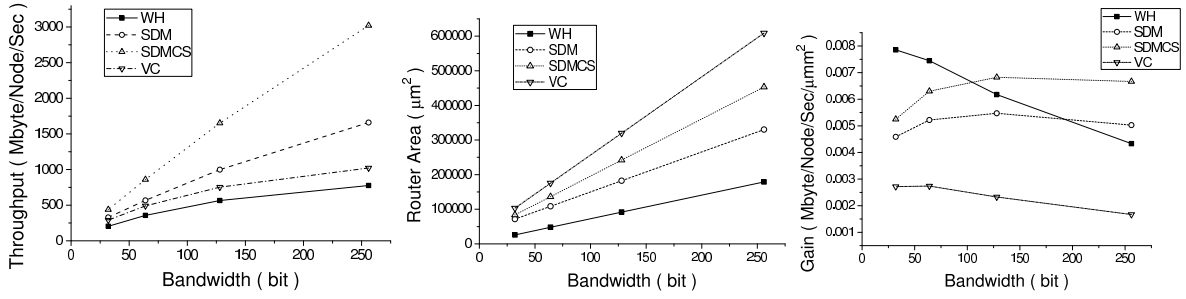
Increasing the buffer length in input buffers alleviates the HOL problem and raises throughput but introduces area overhead. Figure 19 reveals the impact of increasing the buffer length. Both throughput and area overhead rises linearly. Figure 19(c) shows the gain of throughput per area unit. Higher 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 ($L \geq 8$), the SDM router using channel slicing has the best area efficiency. In all cases, VC routers have 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 if the synchronized asynchronous buffer stages are in use. Figure 20 reveals the impact of increasing bandwidth. As expected, the area increases linearly with bandwidth but not throughput. Compared with wormhole and VC routers, SDM routers show steadier throughput increment because their C-element trees are shorter. Using channel slicing has the best throughput increment because all C-element trees are eliminated. Figure 20(c) shows



(a) (b) (c)

Figure 19: (a) throughput, (b) router area and (c) gain of throughput per area with various buffer length ($P = 5, W = 32, M = 4$)



(a) (b) (c)

Figure 20: (a) throughput, (b) router area and (c) gain of throughput per area with various bandwidth ($P = 5, L = 2, M = 4$)

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 128 bits. When the bandwidth of a port is over 128 bits, SDM with channel slicing demonstrates the best area efficiency while VC routers are the worst in all cases.

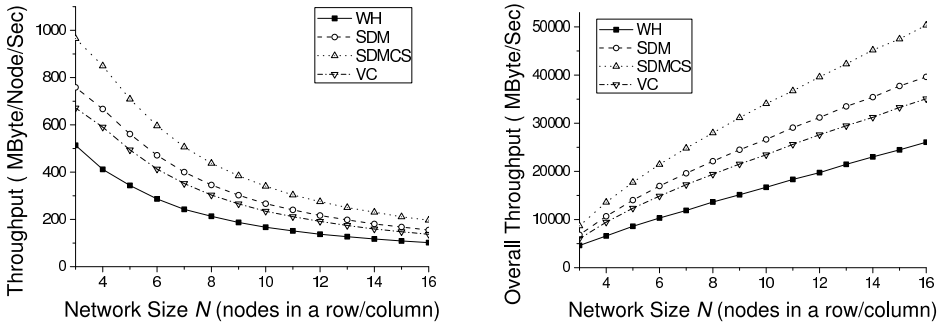
Since both VCs and virtual circuits use the wormhole flow control method, VC routers and SDM routers have similar scalability to network size with wormhole routers under uniform traffic patterns. As described in [4], Equation 28 gives the upper bound on throughput Θ

$$\Theta \leq \Theta_{ideal} \leq \frac{2WB_C}{TN^2} \tag{28}$$

where B_C is the minimal channel count over all bisections of the network and N^2 is the total number of nodes in a mesh network. As B_C in a mesh network is linear with N, the upper bound on throughput and the overall throughput of all nodes in the network can be simplified into

$$\Theta \leq \frac{k}{TN} + C \tag{29}$$

$$\Theta N^2 \leq \frac{kN}{T} + C \tag{30}$$



(a) (b)

Figure 21: Network scalability ($P = 5, W = 32, L = 2, M = 4$)

where k and C are constants. Figure 21 reveals the practical throughput and overall throughput of networks with different sizes. Figure 21(a) verifies the inverse relation between Θ and N, and the overall throughput in Figure 21(b) is approximately linear with N. Note that Equation 30 provides only the upper bound on overall throughput. Although VC routers have larger cycle period than wormhole routers, they show better overall throughput thanks to their capability of resolving HOL problems. After all, all the flow control methods show similar scalability to natwork size.

The number of virtual circuits in an SDM router or 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. Figure 22 demonstrates the latency versus injected traffic performance for SDM and VC routers with various number of virtual circuits and VCs. The estimated area of all these routers are as follows (in unit of μm^2):

SDM(M=2)	38,153	SDM(M=4)	71,713
SDMCS(M=2)	47,661	SDMCS(M=4)	83,626
VC(M=2)	49,896	VC(M=4)	103,250

Both VC routers and SDM routers benefit significantly from the increased number of VCs/virtual circuits. Their throughput increase by 26.6% and 26.8% respectively. On the other hand, SDM routers using channel slicing has a small throughput increment of only 6.7%. The SDM router using channel slicing with two virtual circuits has almost exploited the full throughput capability.

It is also shown in Figure 22, 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, increasing the number of virtual circuits reduces the effective bandwidth of a single virtual circuit. Data are serialized to fit the small bandwidth; therefore, the frame latency rises accordingly.

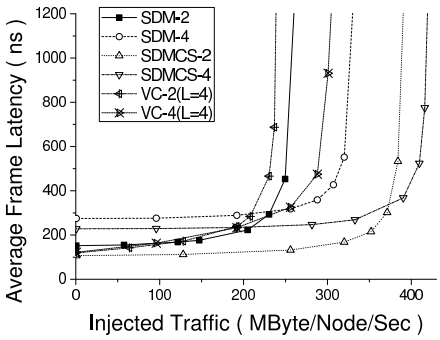


Figure 22: Throughput with various number of virtual circuits or VCs ($P = 5, W = 32, L = 2$)

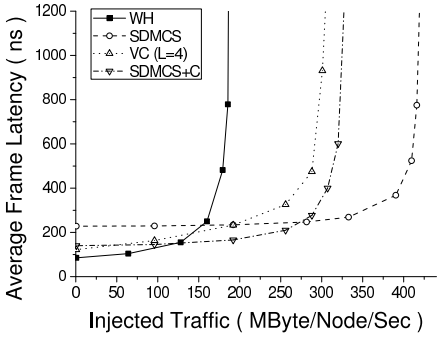


Figure 23: Concurrent frame transmission ($P = 5, W = 32, L = 2, M = 4$)

5.3. Coping with the latency

As shown in Figure 16, SDM routers introduce longer frame latency in low network load because the effective bandwidth of a single virtual circuit is reduced. A tradeoff can be made to reduce this frame latency if the MPSoC system allows a frame to be separated into multiple sub-frames and be transmitted on multiple virtual circuits concurrently.

Assuming every frame has a payload of 64 bytes, the concurrent frame scheme allows every frame to be separated into two sub-frames, each of which contains a 32 bytes payload and an frame sequencer of one byte. Figure 23 shows the latency performance of using the concurrent frame scheme on an SDM router using channel slicing (SDMCS+C). It is shown that the latency in low network load is reduced by 39%. Although it is still longer than the latency of using wormhole routers, it is around the same with that of VC routers (VC routers are with four buffer stages to avoid the credit loop stall). However, sub-frames are sequenced. The concurrent frame scheme introduces extra control overhead, which compromises throughput.

6. 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

frames 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 outperform routers using the VC control flow method and SDM routers using channel slicing show the best throughput performance. 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. The latency can be reduced if a frame is divided into sub-frames and delivered concurrently on multiple virtual circuits.

References

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

[2] J. Sparsø, S. Furber, Principles of Asynchronous Circuit Design — A Systems Perspective, Kluwer Academic Publishers, 2001.

[3] M. Krstić, E. Grass, F. K. Gürkaynak, P. Vivet, Globally asynchronous, locally synchronous circuits: overview and outlook, IEEE Design and Test of Computers 24 (5) (2007) 430–441.

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

[5] T. Felicijan, S. B. Furber, An asynchronous on-chip network router with quality-of-service (QoS) support, in: Proc. of IEEE International SOC Conference, 2004, pp. 274–277.

[6] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, 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] A. Sheibanyrad, Asynchronous implementation of a distributed network-on-chip, Ph.D. thesis, University of Pierre et Marie Curie (2008).

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

[9] J. Bainbridge, S. Furber, Chain: a delay-insensitive chip area interconnect, IEEE Micro 22 (2002) 16–23.

[10] L. A. Plana, S. B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, S. Yang, A globally asynchronous, locally synchronous infrastructure for a massively-parallel multiprocessor, IEEE Design and Test of Computers 24 (5) (2007) 454 – 463.

[11] A. Leroy, D. Milojevic, D. Verkest, F. Robert, F. Catthoor, Concepts and implementation of spatial division multiplexing for guaranteed throughput in networks-on-chip, IEEE Transactions on Computers 57 (9) (2008) 1182–1195.

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

[13] The international technology roadmap for semiconductors, http://www.itrs.net (2007).

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

[15] R. Dobkin, R. Ginosar, C. P. Sotiriou, Data synchronization issues in GALS SoCs, in: Proc. of ASYNC, 2004.

[16] A. Sheibanyrad, A. Greiner, Two efficient synchronous – asynchronous converters well-suited for networks-on-chip in GALS architectures, Integration, the VLSI Journal 4 (1) (2008) 17–26.

[17] E. Rijpkema, K. Goossens, A. Rădulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander, Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip, in: Proc. of DATE, 2003, p. 6.

[18] P. T. Wolkotte, G. J. Smit, G. K. Rauwerda, L. T. Smit, An energy-efficient reconfigurable circuit-switched network-on-chip, in: Proc. of the International Parallel and Distributed Processing Symposium, 2005.

[19] P. Abad, V. Puente, P. Prieto, J. A. Gregorio, Rotary router: an efficient architecture for CMP interconnection networks, in: Proc. of ISCA, 2007, pp. 116–125.

[20] W. J. Dally, Virtual-channel flow control, IEEE Transactions on Parallel and Distributed Systems 3 (2) (1992) 194–205.

[21] T. Bjerregaard, 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.

[22] R. R. Dobkin, R. Ginosar, A. Kolodny, QNoC asynchronous router, Integration, the VLSI Journal 42 (2) (2009) 103–115.

[23] C. Gómez, M. E. Gómez, P. López, J. Duato, Exploiting wiring resources on interconnection network: increasing path diversity, in: Proc. of Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2008, pp. 20–29.

[24] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev, Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers, IEICE Transactions on Information and Systems E80-D (3) (1997) 315–325.

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

[26] M. B. Josephs, J. T. Yantchev, CMOS design of the tree arbiter element, IEEE Transactions on VLSI 4 (1996) 472–476.

[27] K. S. Low, A. Yakovlev, Token ring arbiters: An exercise in asynchronous logic design with Petri nets, Tech. rep., Newcastle University (1995).

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

[29] S. Golubcovs, D. Shang, et al, Multi-resource arbiter decomposition, Tech. rep., Newcastle University (2009).

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

[31] A. Bystrov, D. Kinniment, A. Yakovlev, Priority arbiters, in: Proc. of ASYNC, 2000, pp. 128–137.

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