An Asynchronous Router for Multiple Service Levels Networks on Chip

Dobkin (Reuven) Rostislav, Victoria Vishnyakov, Eyal Friedman, Ran Ginosar
VLSI Systems Research Center, Technion—Israel Institute of Technology, Haifa 32000, Israel
ran@ee.technion.ac.il

Abstract

Networks on Chip that can guarantee Quality of Service (QNoC) are based on special routers that can support multiple service levels. GALS SoCs call for asynchronous NoC implementations, to eliminate the need for synchronization when crossing clock domains. An asynchronous multi-service level QNoC router is investigated. It comprises multiple interconnected input and output ports, and arbitration mechanisms that resolve any output port and service level conflicts. Buffering and credit based transport are enabled, enhancing throughput. A synchronous and an asynchronous routers have been designed, and their performance is compared. The asynchronous router requires less area and enables a higher data rate.

1. Introduction

Interconnect is a very expensive resource within large systems on chip (SoC), consuming area and power and incurring high delays relative to gate delays and clock cycles [1]. Requirements for wide-bandwidth inter-module communications exacerbates the problem, incurring larger area and power costs for the interconnects. In addition, data synchronization problems arise in multi-clock domain SoCs, and operating clocked interconnects becomes increasingly more difficult. Large SoCs are treated as Globally Asynchronous Locally Synchronous (GALS) systems, calling for suitable interconnects beyond conventional synchronous buses. Networks on Chip (NoC) are advocated as a solution for the SoC interconnect problem [2]–[4]. To support varying communication requirements, a Quality-of-Service NoC (QNoC) has been proposed, which performs preemptive routing according to packet priority [5]. Since it is designed to support GALS systems with multiple clock domains, including dynamic scaling of voltage and frequencies per each synchronous module, it is best implemented as asynchronous circuits.

A 2D mesh architecture of QNoC is shown in Figure 1 [5]. The SoC comprises modules and a NoC, consisting of links and routers. All inter-modular communications are carried out in packets; legacy modules (capable only of bus-oriented read/write operations) may require wrappers that handle packet based communications. Packets are partitioned into small flits, which are sent through the NoC using wormhole routing. Each QNoC packet carries a Service-Level (SL) priority tag, related to data communication requirements. In this paper we explore QNoC routers that support four service levels, as defined in [5] (Table 1). The packet consists of three types of flits: a header flit with routing address, body flits and a tail flit, indicating end-of-packet (EOP), as in Figure 2. Each flit contains bits indicating its type and SL.

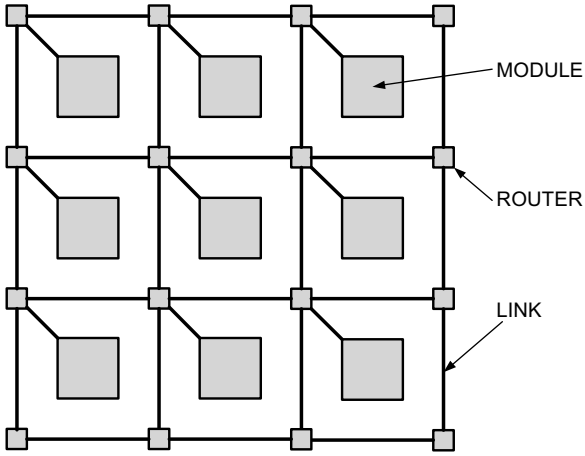

Figure 1: QNoC 2D Mesh Architecture

Table 1: Service Levels Example [5]

Service-Level	Description	Priority
Signaling	Urgent Messages, Short Packets, Interrupts, Control signals requiring low transport latency	Highest
Real-Time	Real-time application packets	
RD/WR	Short memory and register access	
Block Transfer	Long messages and blocks of data	Lowest

Flat 2D mesh QNoC may be inappropriate in some applications. Typically, module placement on SoC strives to minimize long range communications, and blocks that

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 **IEEE**

require high bandwidth interconnect between them are placed close to each other. Thus, most communications are ideally short range. However, long range communications, though usually needing lower bandwidth, cannot be avoided. On the other hand, some nearest-neighbor communications may be optimized by creating custom links that avoid NoC overhead. These observations give rise to hierarchical multi-level NoC architectures that offer different types of links and communication channels for different needs. An example is shown in Figure 3, where three types of links are supported: neighbor, local, and global. The Hierarchical QNoC (HQNoC) is designed to employ the same multiple service level routers, and manage the use of different links through different service levels.

CHAIN [6],[7] proposes a different asynchronous interconnect for NoC. Its CHAINlink protocol is based on 1-of-4 encoding, routers and arbiters. CHAIN provides a flexible framework for NoC, but it is limited to a single service level. An asynchronous router architecture with QoS support was recently presented in [3][8]. Synchronous routers using round-robin arbitration and supporting asynchronous interconnect are presented in [9] [10], though synchronization issues are ignored. Asynchronous packet routers for off-chip networks were presented as early as 1994 [11]. Synchronous NoC routers supporting virtual channels, which could be used to provide multiple service levels, are described in [12] and [13]. Other synchronous routers are discussed in [14]. A synchronous five-port router that supports two service levels (best effort and guaranteed throughput) is described in [15]. The butterfly fat-tree interconnect of [16] may also take advantage of routers described in this paper. NoC wrappers and synchronization issues are discussed in [17]–[22].

We present a single service level QNoC router in Section 2 and reuse the same components in a multi service level router in Section 3. Performance analysis of our design is discussed in Section 4.

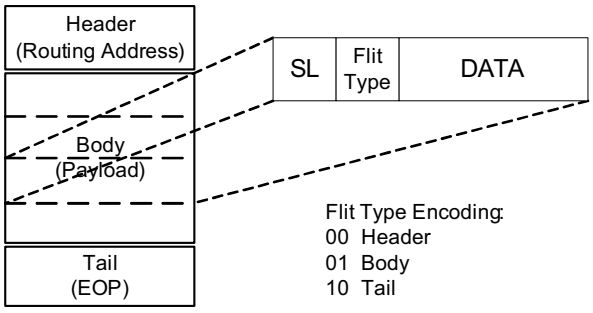

Figure 2: Packet Structure and Flit Format

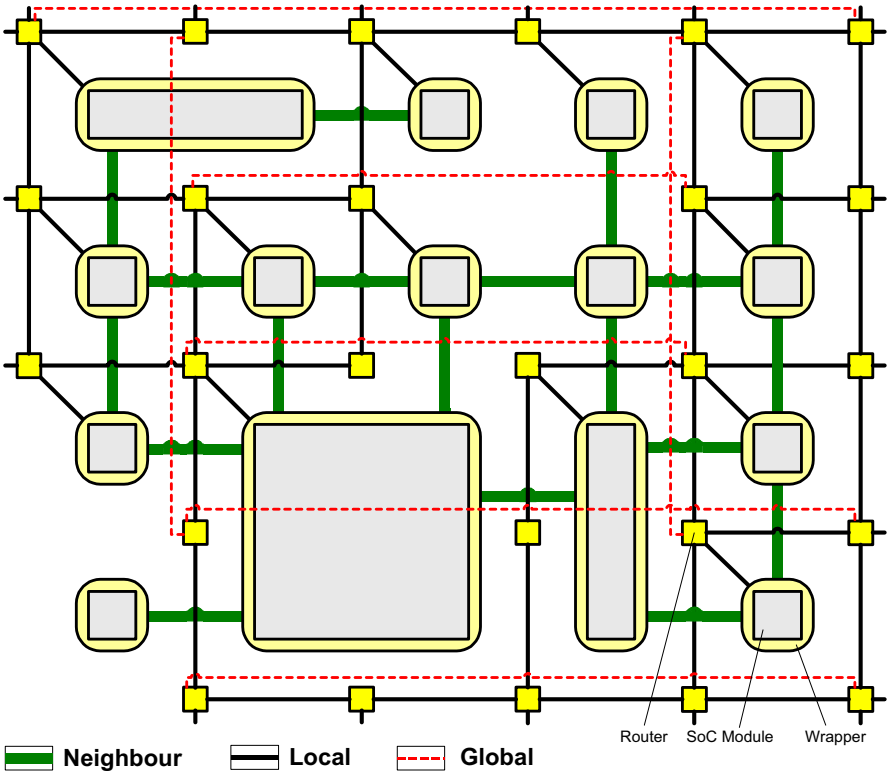

Figure 3: 3-Level Hierarchical NoC Example

2. A Single Service Level (SSL) Asynchronous Router

Routers are the main functional blocks of QNoC, routing flits from an input port (IP) to one of the output ports (OP), according to the routing address and packet priority. The routing is either specified by the source or computed at each router along the route. In computed routing, the packet contains the destination address and each router determines its own switching. One example is X-Y routing for a 2D mesh [5][23], where the packet is first routed along the X dimension and then along the Y dimension towards its destination. When source-specified routing is employed, the packet contains a list of switching indices, providing a switching command for each router [6]. In this paper we employ a simplified version of a source-specified routing: m bits are used to specify switching of an incoming packet to one of 2m possible output ports at each router. The *m* bits are consumed by the router, and the packet needs M×*m* routing bits when M routers are traversed. In the following we describe, as an example, a 5-port router (Figure 4), namely *m*=2.

The bi-directional router interfaces consist each of an input port (IP) and an output port (OP). We consider a router with five interfaces, suitable for a 2D mesh with an additional interface to a SoC Module (as in Figure 1). We assume that a packet coming through an IP does not loop back, and thus each IP is connected to four OPs (Figure 4b) and only two bits are required for switching (Figure 4a). The OPs emit flits according to their arrival order and their priority, as defined by the packet's Service Level (SL). In the rest of this section we discuss the architecture of a single service level router. In the next section we extend this to supporting multiple service levels.

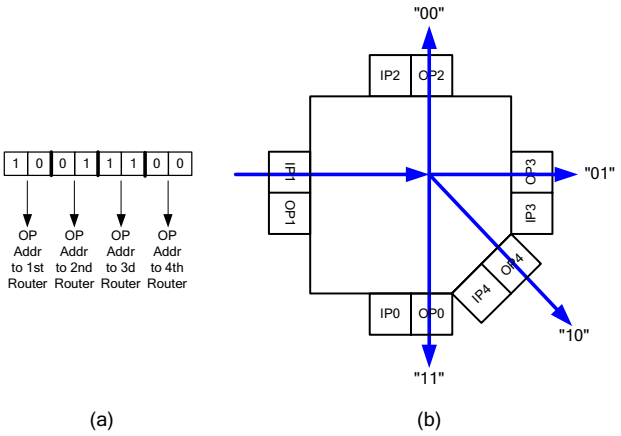

Figure 4: Routing Address from Source to Sink

2.1. Single Service Level Input Port (SSL-IP) Architecture

SSL-IP manages incoming flits that belong to a single service level. The incoming flits are first saved in an internal buffer L (Figure 5), decoupling the external (input) interface and internal processing, and enabling additional flit transmissions. Next, the SSL-IP decodes the flit type (header, body or tail).

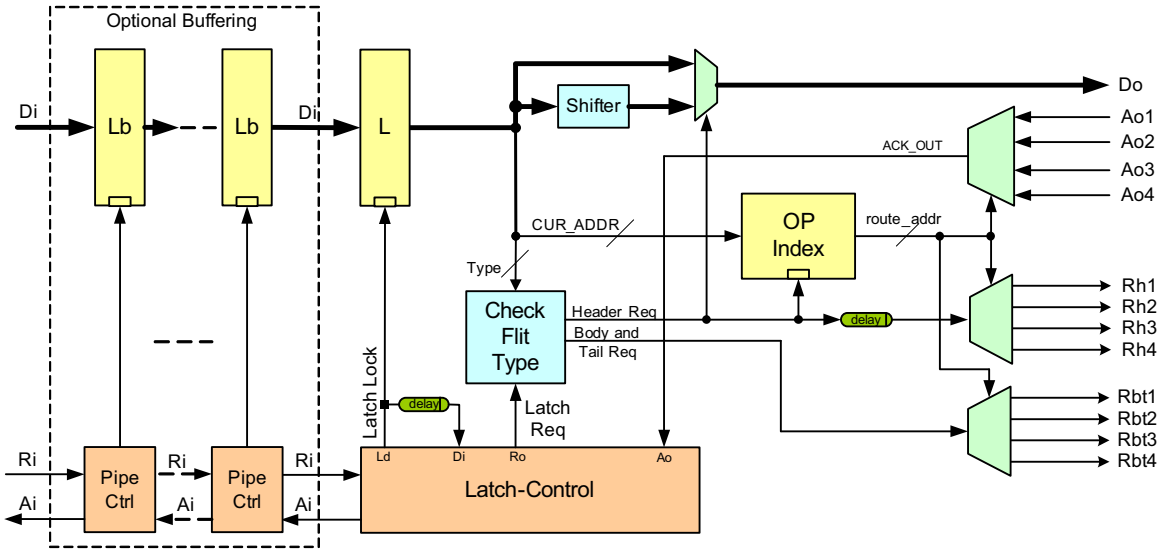

Figure 5: Single Service Level Input Port Architecture

On a header flit, the first two data bits contain the target OP index *i* for the present router. This index is saved in the OP-Index latch, controlling all muxes that select one of four OPs. The index will serve all subsequent flits of the same packet, and will be changed only by the header flit of the next packet. In addition, a shifted version of the header flit is sent out, so that the first two data bits now contain the OP index for the next router. Last, the header is sent out by signaling Rh_i. No processing is required for body and tail flits—they are sent out by signaling the common request Rbt_i to the i^{th} OP. Note that the controller employs asymmetric delay lines to match latch propagation delays.

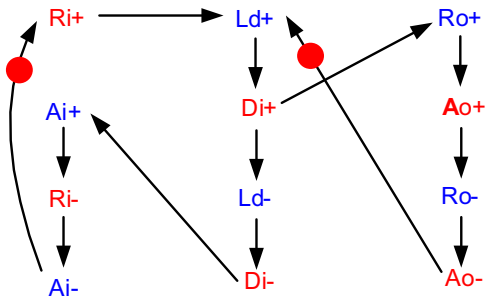

Figure 6: Latch-Control STG

The Latch-Control STG is shown in Figure 6. The external interface (Ri, Ai) is decoupled and is released as soon as the flit has been latched inside the IP (upon Di-). The controller was synthesized using Petrify [24] (see Section 4).

In order to enhance throughput, asynchronous buffer stages may be inserted on the input link (dashed block in Figure 5). The number of buffers depends on throughput and latency requirements. In case of a single service level NoC, the buffers may be spread along the interconnect wires between routers (serving also as repeaters). As explained below, this is different for multi-service level NoCs.

2.2. Single Service Level Output Port (SSL-OP) Architecture

SSL-OP (Figure 7) interfaces four IPs and one external output. It monitors incoming requests and when one is granted it establishes an IP-OP connection and maintains it for the duration of the packet, until receiving a tail-type flit (different packets from the same service-level cannot be interleaved).

The Latch-Control unit latches the selected data in data latch L. Subsequently, it conducts the external handshake with the next router. The unit is identical to the one used in SSL-IP, having same STG as Figure 6.

On a header flit, the current IP index is saved inside the IP-Index latch prior to data latching. Actually, Latch-Control receives no input request (Ri) before the index is latched. After header flit handling, body and flit requests arrive in a mutually exclusive manner. Body and tail flits are immediately sent out to the external interface, through latch L.

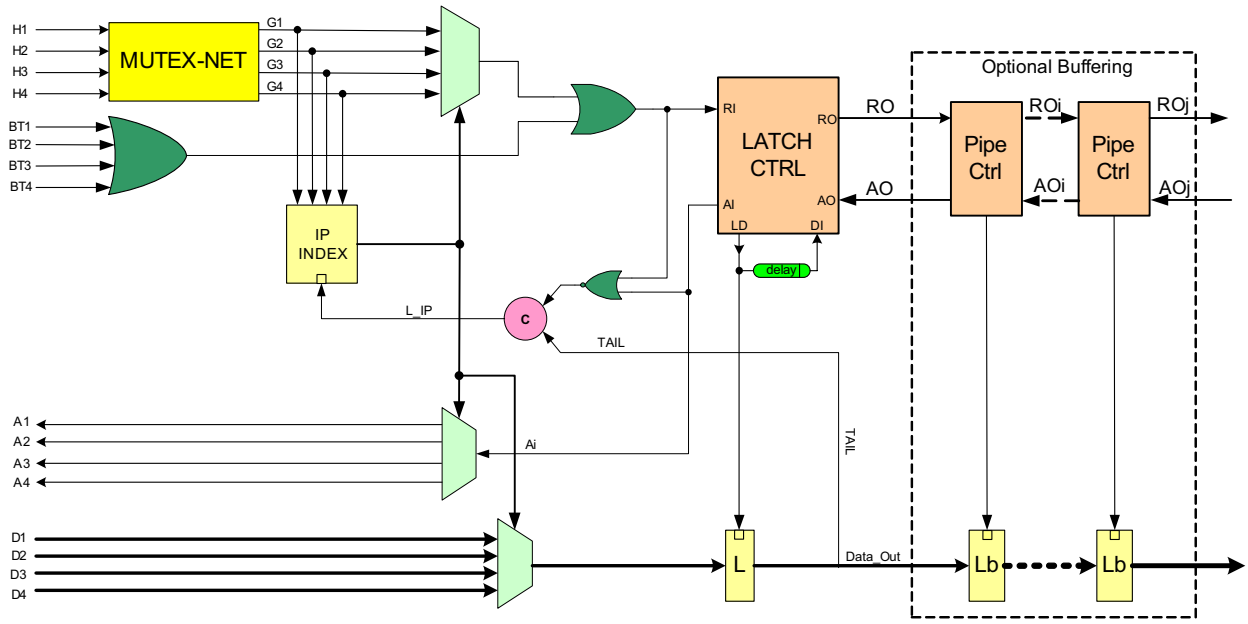

Figure 7: Single Service Level Output Port Architecture

The value inside the IP-Index latch remains unchanged throughout packet transmission, continuously connecting the SSL-OP with the current source SSL-IP. The value is updated by an incoming header-flit and is locked as soon as header-type appears at the output of data latch L, switching the c-element output to low. During body flit transmission the value inside the IP-Index register remains unchanged. Upon a tail-flit, the IP-Index latch becomes transparent allowing the processing of the next packet. The latch becomes transparent only after the port completes the (Ri, Ai) handshake for the tail-flit. This is assured by the NOR gate, keeping the c-element input low during the tail-flit data cycle.

Similarly to the SSL-IP, output buffers can optionally be used to enhance router performance.

Since the router is asynchronous, arrival time of the request signals is unknown, and requests may conflict. Therefore, the four requests should be arbitrated. Note that only header-type requests are arbitrated, since once an IP-OP connection is established, requests from the other IPs are blocked.

Arbitration can be performed using either a tree arbiter (Figure 8b), consisting of three standard two-way arbiters (Figure 8a), or a MUTEX-NET (Figure 9). Both architectures incur similar latency and area. Since MUTEX-NET seems to be slightly faster than the tree-arbiter we use it in our design, and its *fairness* [25] is analyzed below.

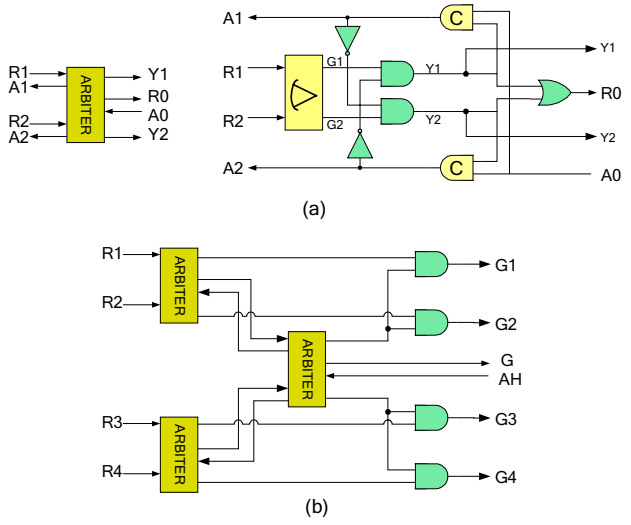

Figure 8: Tree Arbiter

In Figure 9, four requests are mutually excluded by means of a net of six two-input MUTEX elements, arranged in three stages. The latency of the MUTEX-NET is expected to be very low for non-conflicting cases, making this solution fast and effective for the majority of packet transmissions.

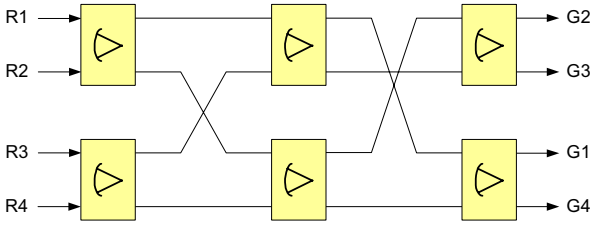

Figure 9: MUTEX-NET

The MUTEX-NET does not always preserve the original order of the incoming requests. For instance, assume the sequence R1+, R2+, R4+, R1-. Clearly, R2+ is blocked by the first stage of MUTEX-NET, while R4 is blocked by the last one. When R1 is released, R2 is blocked again at the second stage by R4, and R4 is granted, even though it came after R2. However, a request is not starved forever inside the net, as follows.

Claim: MUTEX-NET has a bounded blocking time, and a request may be outrun by no more than two later requests.

Proof: Assume that there are four concurrent requests at the input of the first MUTEX-NET stage, say R_i, R_i', R_K, R_K' (the tag means a "pair request," namely a request connected to the same two-input MUTEX of the first stage as the untagged request). Then, at most two of them, say only R_i (worst-case: R_i wins over R_K at the second stage), are granted at the second stage, and only one, R_i, wins at the last (third) stage. When the winner request R_i is released, its pair request R_i' wins at the first stage, blocking any new R_i propagations through the net. Then, there is a race between R_i' and R_K that will end in the worst case by blocking R_K again at the third stage. In this case after R_i' is released, the subsequent request that will be granted is R_K, since by being at the third stage it blocks the pair request R_i. This is the worst-case, and R_K was outrun by two other requests from a different input pair. When at the beginning R_K and R_i both meet only at the third stage, then R_K will be outrun at most by R_i. **QED.**

Conclusions: Given a request R, the following hold:

a. R cannot be outrun by its pair request R' (only in case of concurrency).
b. When blocked by a request from another pair at the second stage, R will always propagate to the third stage once the blocking is released.
c. When blocked by a request from another pair at the third stage, R will win the blocker's pair request after the block is released, thus R is granted next (this is thanks to the blocking of the pair request at the second stage).

In an arbiter, one of the main concerns is *fairness*, which guarantees that a request will be granted after a

bounded number of other requests [25]. Fairness and correctness [26] of arbitration can be improved by using ordered arbiters [27], preserving the closest possible granting order to input arrival, by storing the incoming requests in an internal FIFO. We consider the scheme of Figure 9 as fair enough for our application and use it in the implementation example.

3. Multi-Service Levels (MSL) Asynchronous Router

We now describe the input and output ports that support multiple service levels. Additional bits are added to each flit to identify the service level [5].

3.1. Multi-Service Level Input Port

The QNoC router input port comprises k SSL-IPs (k is the number of service levels). Figure 10 shows a k=4 example. For each incoming flit, the request is applied to only one of the four SSL-IPs, according to the service level. The selected SSL-IP conducts handshake with the input channel asking for data transmission. After the data is latched inside the SSL-IP, a request is sent to the appropriate OP, according to the latched flit's routing address. Additional asynchronous buffering per each service level can optionally be employed, as shown in Figure 10.

3.2. Multi-Service Level Output Port (MSL-OP)

The QNoC multi-service level output port structure is shown in Figure 11. The header requests from the IPs are grouped according to their service level, and conflicts within each service level are resolved using the MUTEX-NET inside the corresponding SSL-OP (see Figure 7).

Flit requests from all service levels enter the static priority arbiter (SPA) [25]. The SPA decides according to service level priority which flit is sent at the next output data cycle. When a service level is granted (G_SL$_i$), the corresponding SSL-OP is connected to the data link and sends one flit through the shared output interface. After sending one flit, control is returned to the SPA, since there could be higher service level flits pending. Priority decision is performed only when the output data cycle is over, thanks to the Gate signal of the SPA. Gating is de-asserted as soon as current cycle is over (on Ao–).

A modified SPA [25] consists of a Request Lock Register (containing the MUTEX elements) and priority logic (Figure 12). When at least one request is sensed, the set of pending input requests are locked in the register, and eventually the highest priority request is granted at the output (G$_i$+). As a result, the Request Lock Register is reset. The C-element holding the grant is released only after the corresponding request goes low. Although fairness of the priority arbiter has recently been improved [28], we employ a modified version of the simpler approach [25], since in our case fairness among service levels is less of an issue, thanks to additional MUTEX-arbitration within each service level.

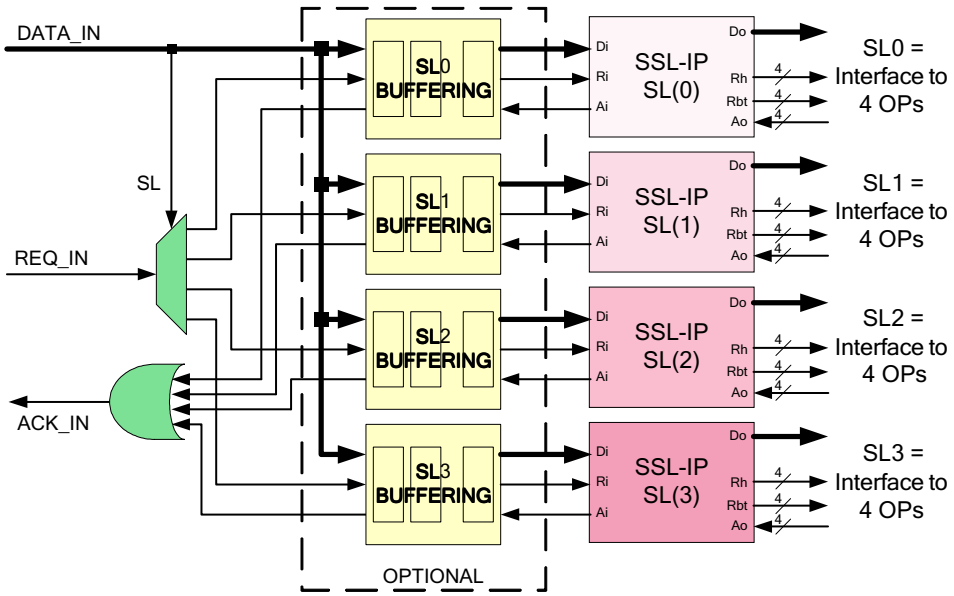

Figure 10: QNoC Multi-Service Levels Router Input Port

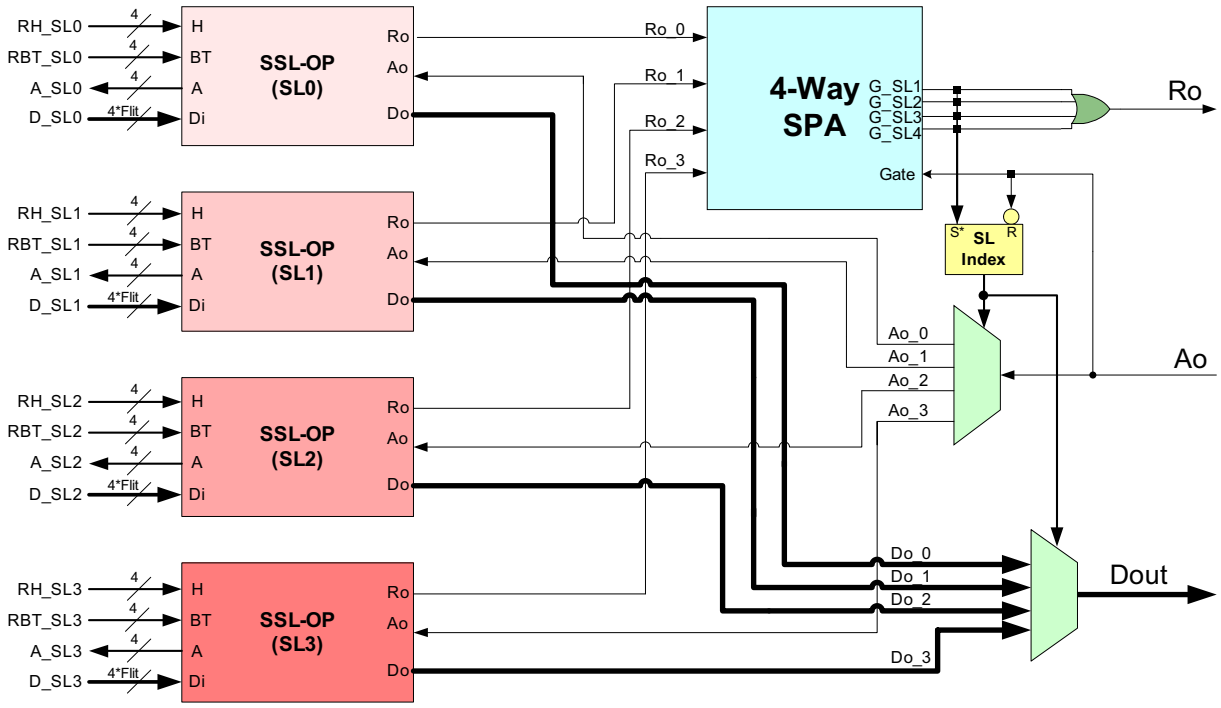

Figure 11: QNoC Multi-Service Level Router Output Port

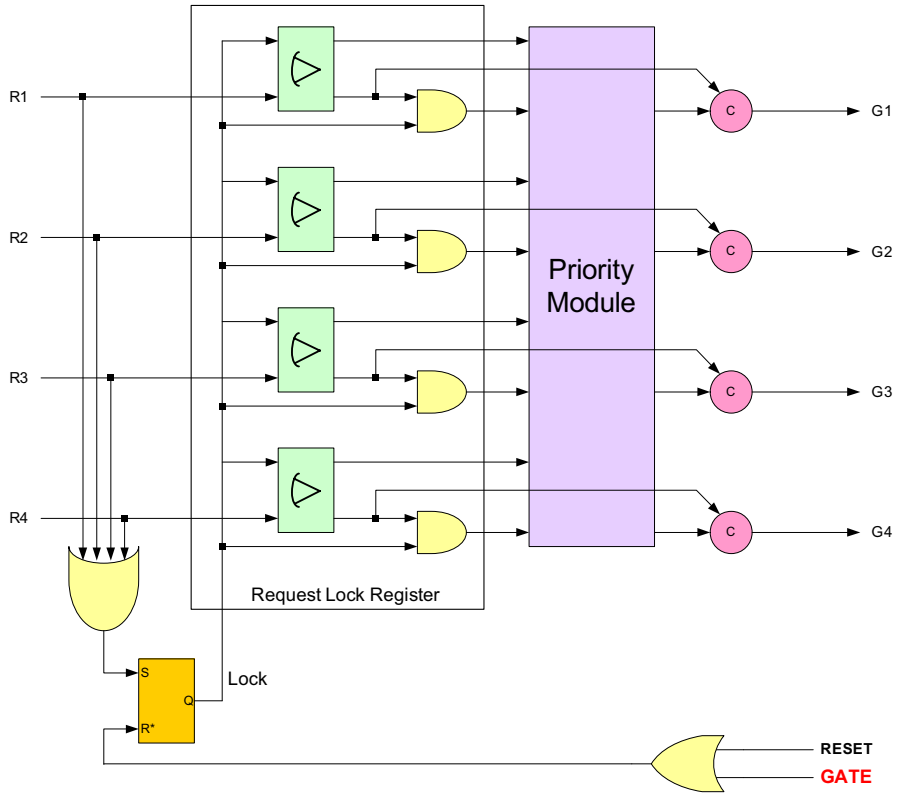

Figure 12: Four-Way Static Priority Arbiter

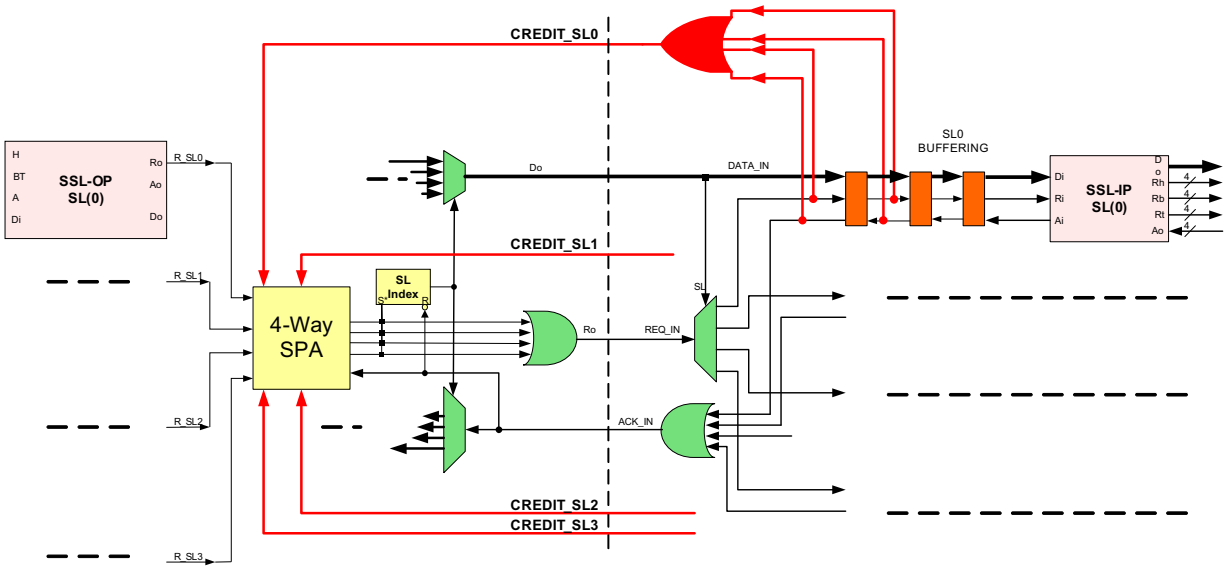

Figure 13: Credit Mechanism in Multi-Service Levels Asynchronous Router

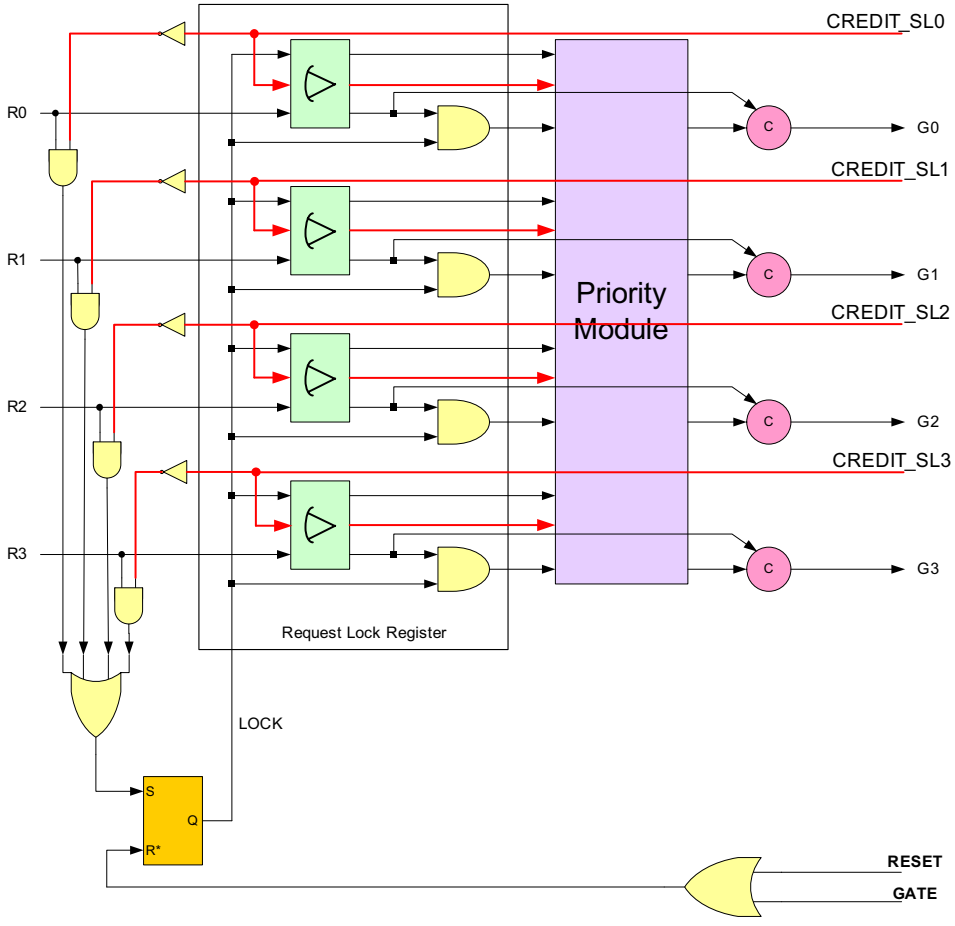

Figure 14: Four-Way Static Priority Arbiter with Credits Support

Table 2: Comparison results for Complete Router for 1- and 4-SL

Parameter	Synchronous Router		Asynchronous Router			Units
	1-SL	4-SL	1-SL, [6]	1-SL	4-SL	
Cell Area	210,000	960,000	-	93,300	470,000	μm^2
Equivalent Gates (2-in NAND)	3,800	17,500	-	1,700	8,500	Gates
Number of transistors	15,200	70,000	~19,000	6,800	34,200	
Number of FFs / Latches	195	880	-	130	620	
Min Latency (Input to Output)	3.3 (1)	3.7 (1)	-	7.6 / 3.9	13.0 / 9.2	ns (CLKs)
Data Cycle	13.2 (4)	14.8 (4)	-	18.0 / 11.9	13.3 / 13.3	ns (CLKs)
Max Data Rate	75.8	67.6	~120	55.5 / 84.0	75.2 / 75.2	Mflits/s
Max Clock Frequency	303.0	270.2	-	-	-	MHz

Notes: (1) Times and rates for async router are specified for header / body&tail flits. (2) Numbers for [6] are estimates.

In a single-service level router, the asynchronous handshake serves also as the "credit" mechanism [5], allowing data transfer only when there is a free buffer space in the next router. In the multi-service level case, another approach must be applied, since the single asynchronous communication channel is shared among all service levels. A free space indication signal is required per each service level (Figure 13). This indication enables requests of the same service level inside the SPA (Figure 14).

4. Performance

The proposed asynchronous QNoC router is compared with a synchronous router of the same functionality. The asynchronous controller was synthesized with Petrify [24] and Synopsis Design Compiler, using a 0.35μm standard cell library. The synchronous router was designed based on the same technology and has a critical path / clock cycle of ~20 FO4 gate delays, matching or outperforming other published results [12]–[15], at a smaller area. Eight-bit flits were assumed for the single service level router, and ten-bit flits for the four service level one. The results are listed in Table 2. The synchronous and asynchronous designs show similar data rate performance, while the asynchronous routers require less area. The area reduction is mostly the consequence of using latches instead of flip-flops. In addition, we have estimated the cost and performance of a single service level router using CHAIN link [6]. The authors reported 120 Mbps per wire data rate for a case without steering. We have assumed that the rate with steering is not much lower. The total area of the five-port CHAIN router was estimated by extrapolating the data in [6], which relates to a 2×2 router.

Note that the reported results depend closely on the choice of protocols and on implementation details. Future research is required to optimize the design, to investigate alternative protocols, and to evaluate bottlenecks. As explained in Sections 2 and 3, data buffers may be added to enhance performance. Buffered routers have not been analyzed yet, and are expected to improve throughput.

While the synchronous and asynchronous routers exhibit similar performance, it should be noted that when the NoC spans multiple clock domains, a multi-link data transfer may incur the additional penalty of multiple synchronization latencies. An asynchronous NoC helps eliminate en-route resynchronizations.

5. Conclusions and Future Research

We have presented QNoC routers for both single and multiple service levels. The single service level router comprises a number of interconnected input and output ports. The output ports arbitrate conflicting requests. Simplified routing mechanism, as well as minimal buffering, are designed to minimize area and power costs at the router. The multi-service level router provides complete functionality for either planar or hierarchical QNoCs. Service levels are arbitrated according to priority, and allow preemption of lower priority transports. Service level specific credit based signaling between adjacent routers provides for minimized blocking and high performance communications. Buffers may be employed as necessary in either single or multiple service level routers.

QNoC for multi-clock domain GALS SoCs naturally benefit from asynchronous interconnects. Our design demonstrates that asynchronous routers require less area than their synchronous counterparts, and incurs a shorter data delay.

Future study is planned for investigating the following issues. The use of buffering must be analyzed to determine their potential for throughput enhancement. Alternative protocols should be studied, as well as other definitions of service levels. Packet and flit formats should also be questioned. Employing different links, such as serial versus parallel links, should also be investigated in the context of different applications. The entire NoC model should be analyzed when used for specific

applications and different usage models and communication requirement mixes.

6. Acknowledgement

We thank Julia Kurolap and Sergey Klimov for helping develop the architecture. Comments of the anonymous reviewers helped significantly to improve this paper. This research was funded in part by Freescale Semiconductor and Semiconductor Research Corporation (1204.001), Israel Ministry of Industry and Trade (Short Range Radio Consortium) and by Intel Corporation.

References

[1] International Technology Roadmap for Semiconductors (ITRS), 2003.
[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," Proc. DAC, June 2001.
[3] J. Duato, S. Yalamanchili, and L. M. Ni. Interconnection Networks: An Engineering Approach. IEEE CS Press, 1997.
[4] W.J. Dally and B. Towles, Principles and Practices of Interconnection Networks Morgan Kaufmann, 2004.
[5] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QoS Architectrure and Design Process for Cost Effective Networks on Chip", J. Systems Architecture, special issue on Networks on Chip, 50(2-3), pp. 105-128, February 2004.
[6] J.Bainbridge and S.Furber, "Chain: a Delay-Insensitive Chip Area Interconnect," IEEE Micro, vol. 22, no.5, pp.16-23, Sep./ Oct. 2002.
[7] W.J. Bainbridge, L.A. Plana and S.B. Furber, "The Design and Test of a Smartcard Chip Using a CHAIN Self-timed Network-on-Chip," Proc. DATE, Feb. 2004.
[8] T.Felicijan, S.B.Furber, "An Asynchronous On-Chip Network Router with Quality-of-Service (QoS) Support," Proc. of IEEE International SOC Conference, Santa Clara, CA, Sept. 2004, pp. 274-277.
[9] N. Banerjee, P. Vellanki and K.S. Chatha, "A Power and Performance Model for Network-on-Chip Architectures," DATE 2004.
[10] P. Vellanki, N. Banerjee and K.S. Chatha, "Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures," Proc. GLSVLSI'04, Boston, USA, pp. 45-50, 2004.
[11] I.M. Nedelchev and C. R. Jesshope, "Basic building blocks for asynchronous packet routers," Proc. of Fourth Great Lakes Symposium on 'Design Automation of High Performance VLSI Systems', pp. 184-187, March 1994.
[12] L. Peh and W.J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," 7th Int. Symp. High-Performance Computer Architecture (HPCA), Jan 2001.
[13] R. Mullins, A. West and S. Moore, "Low-Latency Virtual Channel Routers for On-Chip Network," Proc. 31st Int. Symp. Computer Architecture, 2004.
[14] H. Wang, L.S. Peh and S. Malik, "Power Driven Design of Router Microarchitectures in On-Chip Networks," Proc. MICRO-36, 2003.
[15] E. Rijpkema, K. Goossens et al., "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," IEE Proc.-Comp. Digit. Tech., 150(5), 294-302, Sept. 2003.
[16] C. Grecu, P.P. Pande, A. Ivanov and R. Saleh, "A Scalable Communication-centric SoC Interconnect Architecture," Proc. 5th Int. Symp. Quality Elect. Design, 343-348, 2004.
[17] R. Dobkin, R. Ginosar and C. P. Sotiriou, "Data Synchronization Issues in GALS SoCs", Proc. ASYNC, April 2004.
[18] J. Kessels, A. Peeters, P. Wielage and S.-J. Kim, "Clock Synchronization through Handshake Signalling," Proc. ASYNC, pp. 59-68, March 2002.
[19] T. Villiger, H. Kaeslin, F.K. Gürkaynak, S. Oetiker and W. Fichtner, "Self-Timed Ring for Globally-Asynchronous Locally-Synchronous Systems," Proc. ASYNC, pp. 141-150, April 2003.
[20] J. Muttersbach, T. Villiger and W. Fichtner, "Practical Design of Globally-Asynchrounous Locally-Synchronous Systems," Proc. ASYNC, pp. 52-61, April 2000.
[21] S. Moore, G. Taylor, R. Mullins and P. Robinson, "Point to Point GALS Interconnect," Proc. ASYNC, April 2002.
[22] S. W. Moore, G. S. Taylor, P. A. Cunningham, R. D. Mullins and P. Robinson, "Self-Calibrating Clocks for Globally Asynchronous Locally Synchronous Systems," Proc. Int. Conf. Computer Design (ICCD), 2000.
[23] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "Cost considerations in Network on Chip", Special issue on Networks on Chip, Integration - The VLSI Journal, 2004.
[24] 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, E80-D(3), pp. 315–325, 1997.
[25] A. Bystrov, D. Kinniment and A. Yakovlev, "Priority Arbiters," Proc. ASYNC, pp. 128-137, April 2000.
[26] C. H. (Kees) van Berkel and C.E. Molnar, "Beware the Three-Way Arbiter," IEEE Journal of Solid-State Circuits, 34(6), pp. 840-848, June 1999.
[27] A. Bystrov and A. Yakovlev, "Ordered arbiters," Electronics Letters, 35(11), pp. 877-879, 27th May 1999.
[28] T. Felicijan, J. Bainbridge and S. Furber, "An Asynchronous Low Latency Arbiter for Quality of Service (QoS) Applications," Proc. Int. Conf. Microelectronics (IMC), Cairo, Egypt, pp. 123-126, Dec. 2003.