Power and Area Efficient Design of Network-on-Chip Router Through Utilization of Idle Buffers.

Khalid Latif[1,3], Tiberiu Seceleanu[2], Hannu Tenhunen[1,3]
[1]Department of Information Technology, University of Turku, Finland
[2]ABB Corporate Research, Västerås, Sweden
[3]Turku Centre for Computer Science (TUCS), Finland
Khalid.Latif@utu.fi, Tiberiu.Seceleanu@se.abb.com, Hannu.Tenhunen@utu.fi

Abstract

Network-on-Chip (NoC) is the interconnection platform that answers the requirements of the modern on-Chip design. Small optimizations in NoC router architecture can show a significant improvement in the overall performance of NoC based systems. Power consumption, area overhead and the entire NoC performance is influenced by the router buffers. Resource sharing for on-chip network is critical to reduce the chip area and power consumption. Virtual channel buffer sharing by other router ports has been proposed to enhance the performance of on-chip communication. We approach the router architecture optimization by utilizing the idle buffers instead of increasing the number and size of buffers for desired throughput.

1 Introduction

Ever-increasing requirements on electronic systems are one of the key factors for evolution of the integrated circuit technology. Multiprocessing is the solution to meet the requirements of upcoming applications. Multiprocessing over heterogeneous functional units require efficient on-chip communication [11]. Network-on-Chip (NoC) is a general purpose on-chip communication concept that offers high throughput, which is the basic requirement to deal with complexity of modern systems. All links in NoC can be simultaneously used for data transmission, which provides a high level of parallelism and makes it attractive to replace the typical communication architectures like shared buses or point-to-point dedicated wires. Apart from throughput, NoC platform is scalable and has the potential to to keep up with the pace of technology advances [2]. But all these enhancements come at the expense of area and power. In the RAW multiprocessor system, interconnection network consumes 36% of the total chip power [20].

A typical NoC system consists of processing elements(PEs), network interfaces (NIs), routers and channels. The router further contains switch and buffers. Buffers consume the 64% of the total node (router + link) leakage power for all process technologies, which makes it the largest power consumer in any NoC system [22]. Moreover, buffers are dominant for dynamic energy consumption [15]. It is better to transmit packets instead of storing them because more power consumption is expected in storing them as compared to the transmission [17]. Thus, reduction in number and size of buffers with increase in utilization affect the system performance and impact area and power efficiency.

Related work. Buffer management is not a recent issue but still needs attention. This comes as a performance improvement to utilize the unused buffers at some time instant. Lately, buffer sharing has been analyzed and compared to the existing router architectures.

Lan et. al [23] addresses the buffer utilization by making the channels bidirectional and shows significant improvement in system performance. But in this case, each channel controller will have two additional tasks: dynamically configuring the channel direction and to allocate the channel to one of the routers, sharing the channel. Also, there is a 40% area overhead over the typical NoC router architecture due to double crossbar design and control logic. We approach the problem with very simple control logic. There is only one output crossbar and a set of 2×1 input multiplexers to share the buffer between two input ports (distributed approach).

Soteriou et. al [19] introduced distributed shared buffer (DSB) NoC router. The proposed architecture shows a significant improvement in throughput at the expense of area and power due to extra crossbar and complex arbitration scheme. Our approach delivers the same performance without any extra crossbar and memory bank.

Coenen et. al [13] developed an algorithm to optimize size of decoupling buffers in network interfaces. The buffer size is proportional to the maximum difference between the number of words produced and the number of words consumed at any point in time. This approach showed significant improvement in power dissipation and silicon area. The buffer size can be further optimized by considering the idle time of buffer. If some buffer is idle at some time instant, it can share the load of neighboring input channel and thus increase the utilization of existing resources with a small control logic.

Kodi et. al [3] illustrates the impact of repeater insertion on inter-router links with adaptive control and eliminating some of the buffers in the router. The approach saves appreciable amount of power and area without significant degradation in the throughput and latency. But there is still some scope to increase the buffer utilization inside the router by using the architecture which we propose here.

Neishabouri et. al [14] propose the router architecture with *Reliability Aware Virtual Channel* (RAVC). In this approach, more memory is allocated to the busy channels and less to the idle channels. This dynamic allocation of storage shows 7.1% and 3.1% latency decrease under uniform and transpose traffic patterns respectively at the expense of complex memory control logic. Though this solution is latency efficient but not area and power efficient, which was not discussed by the authors.

The remainder of this paper is organized as follows. Section 2 presents the motivation and requirements for proposed router architecture. Section 3 explains our proposed architecture and how it can contribute to reduce the area and power consumption without affecting throughput. It also explains, how the proposed architecture was implemented. Finally experimental results and conclusions are drawn.

2 Motivation

This section presents the requirements for improvement in currently available router architecture, degree of utilization for available resources in current NoC router architectures and previous research for resource utilization to enhance the NoC system performance regarding area, power and latency.

Different on-chip communication platforms have been purposed to deal with system complexity, like NoC and segmented bus architectures. But each has its own advantages, disadvantages and limitations. NoC uses distributed control but on other hand segmented busses are controlled by hierarchy of arbiters. In certain situation, If a dedicated link between two processing units is needed, NoC platform will require two routers and a physical communication channel. This increases design complexity, communication overhead and synchronization issues. Thus NoC provides higher throughput at the expense of extra power consumption and silicon area. On other hand, segmented busses such as the *SegBus* [4] provides an optimized solution for such cases by placing both processing elements in same segment during *place tool* step [6, 24]. The *Segbus* is an optimal solution for applications, where more point-to-point communication is needed.

As the NoC design complexity rises, more communication mechanism issues are raised as well. Wormhole flow control have been proposed to reduce the buffer requirement and enhance the system throughput. But on other hand, one packet may occupy several intermediate switches at the same time. In typical NoC architectures, when a packet occupies a buffer for a channel, the physical channel cannot be used by other channels, even when the original message is blocked [11]. This introduces the problem of deadlock and livelock in wormhole scheme.

Virtual Channels (*VCs*) are used to avoid deadlock and livelock. Fig.1 shows a typical virtual channel router architecture [16]. Virtual channel flow control exploits an array of buffers at each input port. By allocating different packets to each of these buffers, flits from multiple packets may be sent in an interleaved manner over a single physical channel. This improves both throughput and latency by allowing blocked packets to be bypassed.

The drawback of using *VCs* stands in a more complex control protocol, as data corresponding to different messages which is multiplexed on the physical channel must be eventually separated [11]. Another important issue that needs attention is the utilization tradeoff. *VCs* are proposed to increase the utilization of physical channels. By inserting the *VC* buffers, we increase the physical channel utilization but utilization of inserted VC buffers is not considered. It can be observed that if there is no communication on some channel at some time instant and at the same time, neighboring channel is overloaded, free buffers of one channel cannot contribute for congestion control by sharing the load of neighboring channel. Adaptive routing technique provides a solution to these issues but introduces some other problems like packet reordering.

A well designed network exploits available resources to improve performance [8]. So, a tradeoff between system performance and resource utilization is needed. Our motivation and innovation is to propose a router architecture with enhanced utilization of *VC* buffers without affecting the utilization of physical channel to reduce system latency, power consumption and silicon area.

3 The Proposed Router Architecture

The proposed architecture utilizes the unused channel buffers instead of increasing the number and size of buffers. Sharing of all the input buffers among all the input ports can provide the best utilization of buffers but this increase the size of input crossbar and thus the complexity of control logic. However, this is not an area, power consumption and latency efficient solution. We approach the problem by

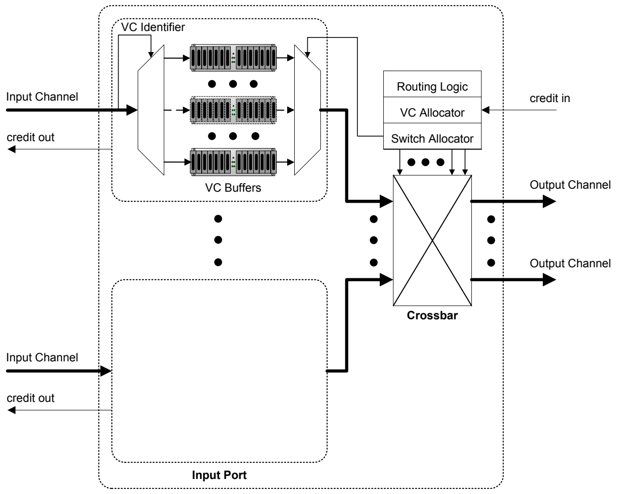

Figure 1. Typical Virtual Channel Router Architecture.

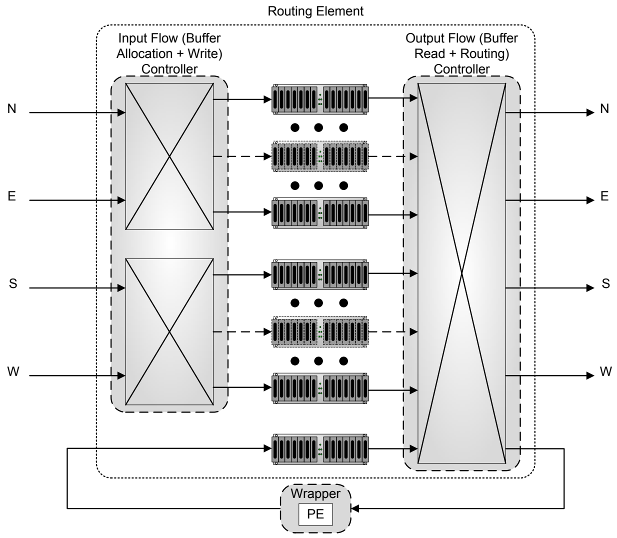
Routing Element
Input Flow (Buffer Allocation + Write) Controller
Output Flow (Buffer Read + Routing) Controller
N
E
S
W
N
E
S
W
Wrapper
PE

Figure 2. System Level specification of Router Architecture.

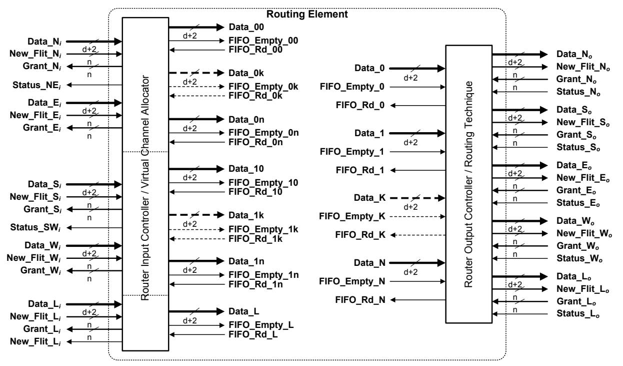

Figure 3. System Level Input and Output Controllers Signalling Specifications of Router Architecture.

sharing *VC* buffers between two ports only. In this case, crossbar size is not so big and control logic is very simple. We provide, thus, a good compromise between control logic complexity, silicon area and the utilization figures.

We adopt here a hybrid control logic approach, which uses distributed control logic implemented by hierarchy of arbiters. Input and output control logics are completely independent of each other and work according to the status of channel buffers. It is a well known fact that routers in NoC system, specifically buffers inside routers consume a big fraction of power and area as discussed in section 1.

The proposed architecture, sharing of buffers between neighboring input ports, is shown in Fig.2. The router architecture can be divided into two parts, the Input and the Output. The Input part is responsible for buffer allocation and receiving the packets from neighboring routers. The Output part computes the route and transmits the packets accordingly. Within the router, the Input and the Output parts do not communicate at all. Both parts simply write and read from the buffers. Fig.3 shows the signals of both parts to write and read from buffers and also the external interface of router.

As the level of system integration in Systems on Chip (SoCs) began to rise, system designers faced the need to reuse pre-designed and pre-verified computation units across different platforms and with different communication architectures. Therefore, the need for effective plug-and-play design styles pushed the development of standard interface sockets, allowing to decouple the development of computational units from that of communication architectures [9].

A wrapper provides the abstraction between PE and the interconnection platform. The wrapper architecture can be divided into two parts: the Network Interface (NI) and the PE Interface (PI) as shown in Fig. 4. NI is a generic interface, which needs to be standardized. Main tasks of the NI are packetization and de-packetization of data. Different services can be introduced in NI architecture, such as multicasting and error monitoring. The queuing buffer for the PE can be considered as the part of NI on input port of the switch. Thus each PE has its own dedicated buffer, which simplifies the control logic and enhance the throughput without any area overhead. PI is the core specific interface, which acts as a clock synchronizer. Different services can be introduced in the PI architecture, such as thermal monitoring and power safe mode, while the core is not in use. Due to this abstraction and synchronization, the NoC platform can be seen as a plug and play platform. This helps

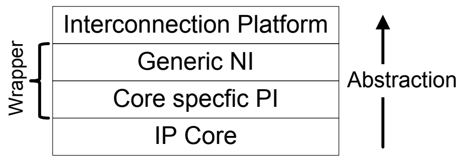

Figure 4. NoC Wrapper Structure.

to reduce the design time, and heterogenous PEs can be integrated to the system.

3.1 Packet Format

Data is packetized by the NI and it is injected into the network according to the format shown in Fig. 5. De-packetization is also the job of NI as explained above. Header flit contains the source address (SA), the operational code (OP), the priority level (PR) and the destination address (DA), as briefly described below:

SA. This field contains the address of the packet source (or request initiator for control signal communication). Devices on the platform are identified by unique numbers, which are used for addressing as well.
OP. The *OP* signal is used to identify the *Operation* to be done on the packet, if destination node PE offers multiple operations.
PR. The *PR* field is used for buffer allocation, when only one buffer is available and both neighboring routers are requesting the buffer allocation or in similar situation for output port allocation, the router with packet of higher *Priority* level wins the allocation.
DA. This field contains the address of the targeted device.
BOP. Beginning of Packet (BOP) identifies the header flit. Buffer allocation unit works on this signal and allocates the appropriate buffer. Output port allocator also works on BOP signal and computes the path according to routing policy to allocate the output port.
EOP. End of Packet (EOP) signal is used to mark the allocated buffer as free and available for next allocation. Similarly, Output port controller knows that packet transmission has been completed. EOP is also a mark for tail flit.

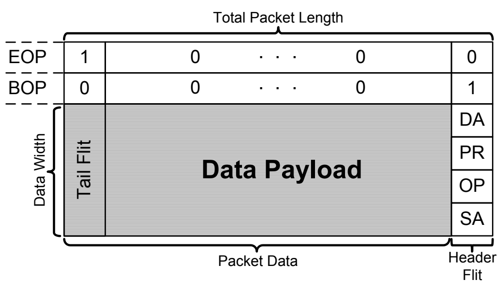

Figure 5. Packet Format.

3.2 The Input Controller and Buffer Allocation

The main contribution of our suggested architecture stands on the input side, where *VC* buffers are shared between neighboring input ports. Each *VC* buffer is shared among two input ports except the buffer dedicated to the local PE. In this section, two input ports are used to explain the complete operation. The control of two ports which share the VC buffers is completely independent from the other input control logic. Internal architecture of input side for two neighboring ports, sharing the *VCs* is shown in Fig.6. Distributed control logic is used to reduce the latency and power consumption.

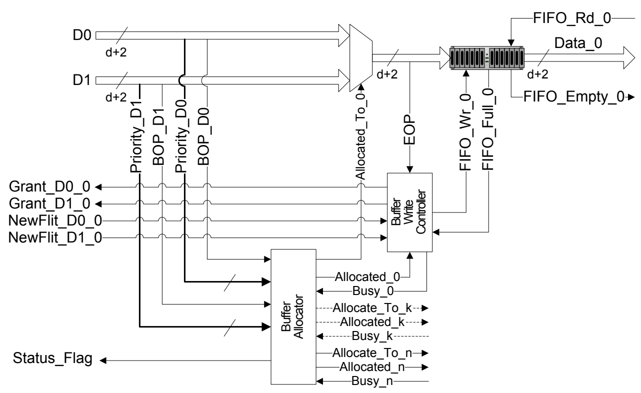

Figure 6. Architecture of Input part of Router.

The *Input control* operation can be divided into several phases, as follows.
***Buffer Allocation*:** The *Buffer allocator* receives the requests from neighboring routers for new buffer allocation in the form of *BOP* signal. The *Buffer allocator* ”knows” the status of all the buffers and allots the unallocated buffer to the requesting router. If only one buffer is available and both routers are requesting for the buffer, the router with higher priority value (*PR*) in header flit is allowed to use the buffer for packet transmission as mentioned in section 3.1.
***Local Signalling*:** After the allocation, the corresponding *Buffer write* controller is informed by raising the *Allocated* signal from buffer allocator to the buffer write controller. In response, the Buffer write controller raises the *Busy* signal. At the same time, the Buffer allocator signals the corresponding multiplexer by *Allocated_To* signal, to which input, corresponding buffer has been alloted.
***Packet Receive*:** After local control signalling, the Buffer write controller signals the *Grant* to the corresponding neighbor router. The latter uses the *Grant* signal as VC identifier for requested packet and always raises the equivalent *NewFlit* signal, while transmitting flits for that packet.

Buffer De-Allocation**:** The *Busy* signal from the Buffer write controller goes down upon arrival of tail flit marked by *EOP* signal as explained in section 3.1. After receiving *Not Busy* signal from the Buffer write controller, the Buffer allocator marks the corresponding buffer as free. If not already raised, the *Status_Flag* is raised.
Status Flag**:** The *Status_Flag* signal is the logical *AND* operation of all the *Busy* signals from buffer write controllers. When all buffers have been allocated, the *Status_Flag* is raised to inform the neighboring routers that no buffer is available for transmission of new packets. If at least one buffer is available for allocation, the *Status_Flag* is in down state.

The *Buffer allocation* unit works only when the *BOP* signal is received. Once the buffer has been tied to the requesting router, the Buffer allocator goes into the sleep mode to save power. It does not consume power until the next BOP signal is received. Similarly, the Buffer write controller works only after receiving the *Allocated* signal from buffer allocator and goes in sleep mode at the *EOP* signal.

3.3 The Output Controller and Routing Algorithm

The *Output* part is modeled by a typical N$\times$5 crossbar, where N is the total number of buffers in the router including the buffer dedicated to local PE. The internal architecture of the output crossbar is shown in Fig. 7. The crossbar size can be customized according to the topology requirements. Here we consider a mesh topology, for deciding the number of I/O ports for the router. Distributed control logic was used here as well. There is one central controller which is the key part of the router. The central output controller decides the routing policy and computes the route for the packet. The port controller controls the packet transmission and the communicates with the destination router, without involving the central controller. It also decides the flow control as well. A worm-hole flow control was used, which makes efficient use of buffer space as small number of flit buffers per VC are required [21].

The *Output controller* operation can be divided into several steps, as follows.
Route Computation**:** The central controller senses the buffers and, on *BOP* signal, computes the route for the packet by using *DA* field as explained in Section 3.1.
Local Signalling**:** After computing the route, the corresponding output port is assigned to the packet. Then central controller informs the corresponding port controller by the signal *New_Buffer_Allocation* which means that the new packet has been allocated on this port for transmission. In parallel, the central controller sends the buffer ID containing the packet to the port controller and FIFO read logic block. After that, the central controller marks that buffer ID as *allocated* internally. It does not read the *BOP* signal for a buffer which is marked *allocated.*
Data Read and Transmission on Output Ports**:** The *Port controller* generates the read signal for that buffer. If the FIFO empty signal is high, the *port controller* does not generate the read signal for that buffer. The FIFO read logic is required to avoid multiple drivers problem for buffer reading. The FIFO read controller keeps track of allocation and allows only the appropriate port to read the buffer.
De-Allocation**:** When the *EOP* bit is high, the port controller marks that buffer as free and sends the corresponding buffer ID back to the central controller, to mark it as *unallocated.* After that, the *central controller* starts checking the *BOP* bit for route computation and portal allocation.

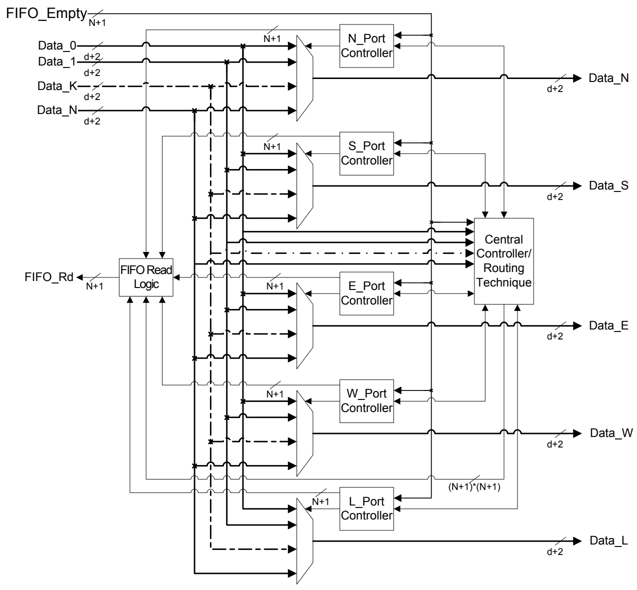

Figure 7. Architecture of Output part of Router.

Communication Between Port Controller and Central Controller: Communication signaling between the central controller (routing element) and the port controller is shown in Fig.8. The FIFO read logic is a combinational logic module intended to avoid multiple driver error. The *central controller* computes the route for the incoming header flit and sends the corresponding output port ID to the port controller by the signal *Buffer_Allocation_ID_N.* In parallel, the central controller raises the signal *New_Buffer_Allocation_N* to inform the port controller that a new buffer has been allocated to it for transmission. After transmitting the tail flit, the *port controller* sends back the *Buffer ID* for deallocation by the signal *Buffer_Deallocation_ID_N.* After that, central controller is waiting for next BOP signal from that buffer for next output port assignment to incoming packets. *Port_Status_N* is raised, when corresponding port cannot make more assignments.

Architecture⇒ Resource⇓	Typical NoC	Normal BiNoC	Reduced BiNoC	Distribute Shared Buffer NoC (DSB-160)	Shared Virtual Channel NoC
Total Number of Buffers	5	10	5	10	5
Buffers/ Direction	1	2	1	1*	1
Total Channels	5-in 5-out	10-inout	5-inout	5-in 5-out*	5-in 5-out
Channels/ Direction	1-in 1-out	2-inout	1-inout	1-in 1-out*	2-in 2-out
Each Buffer Size	32 flits	16 flits	32 flits	16 flits	32 flits
Total Buffer Size	160 flits	160 flits	160 flits	160 flits	160 flits
Crossbar	5X5	10X10	5X5	2 (5X5)	5X5

* 5 memory banks are not considered while making the comparison as only one flit can be written into and read from a middle memory.

Table 1. Comparison with existing NoC router architectures

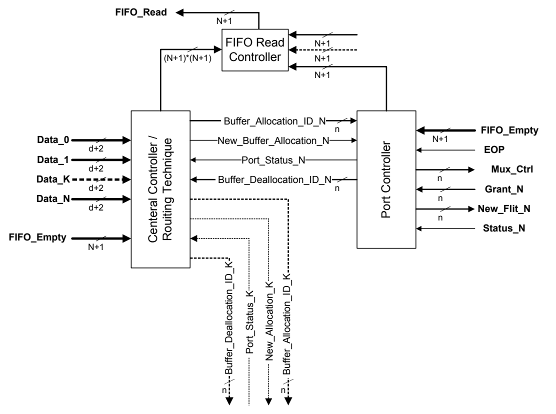

Figure 8. Output Control Architecture.

3.4 Comparison with Existing Architectures

Tthe table. 1. shows a comparison of proposed architecture with existing router architectures. The typical NoC represents a conventional NoC router, which use unidirectional channels to communicate with neighboring routers. Thus two channels are required between two neighboring routers for two way communication. Normal BiNoC has two bidirectional channels, which can be used according to the requirements to communicate in any direction [23]. Reduced BiNoC has only one bidirectional communication channel. This channel can be used for communication in any direction but two way communication is not possible at the single time instant as explained by [23]. In shared VC architecture, all channels are unidirectional but idle buffers can be used by the neighboring port to control congestion. As compared to the BiNoC architecture with 10-in-out channels, the shared VC approach provides 5-input and 5-output channels. Practically, when the BiNoC system is fully loaded, the router will be configured as 5-input and 5-output channel router, which is the same as with the shared NoC architecture. On the other hand, shared VC router can provide 2-input and 2-output channels per direction (if needed), which is double than with the BiNoC architecture.

The DSB-175 and DSB-300 router architectures have already been explained by [19]. To make the exact comparison, we define DSB-160 architecture in accordance with [19]. The DSB-160 is the router with 160-flits of aggregate buffering. The buffers are divided between 5 middle memory banks with 16-flit buffers per bank and aggregate 80-flit input buffers comprising one 16-flit buffer at each input port. As for other architectures, only one buffer per input port or 5 input buffers have been used, VCs are not considered for DSB-160 router.

Another issue to be addressed here is scalability. The number of VC buffers can be selected according to the application and topology requirements for the proposed architecture. To insert a new VC, the buffer and a controller are needed without any modification in existing logic. To insert a new buffer in BiNoC architecture, a separate buffer allocator is required.

3.5 Implementation

The suggested prototype was implemented on Altera Stratix-II FPGA board containing the device EP2S180 [1]. EP2S180 device contains 143,520 ALUTs, 938,3040 bits on chip memory and 12 PLLs. This device was selected for synthesis to make the exact comparison with results provided by [5]. Similarly, 256-flit FIFO depth with 34 bit data width including BOP and EOP bits were used for comparison purposes, otherwise FIFO depth and width are parameterized and values according to the requirement can be selected. A 32-flit buffer size was used for comparison purposes in section 3.4. We used two buffers shared between two ports whereas one buffer was dedicated for each port by [5]. Otherwise, any number of buffers can be shared among neighboring two ports as virtual channels.

4 Experimental results

The proposed architecture showed a significant reduction in silicon area and power consumption, as compared to the architecture described in [5]. The area results are depicted in Table.2. As the buffers can be shared by a couple of input ports in our architecture, significant improvement in buffer utilization was also achieved with limited resources.

Router Type	Total Logic Elements		Dedicated Logic Registers		Block Memory Bits	
	Shared Virtual Channel NoC	NoC Prototype by [5]	Shared Virtual Channel NoC	NoC Prototype by [5]	Shared Virtual Channel NoC	NoC Prototype by [5]
5-Ports	519	920	182	230	43520	43520
4-Ports	498	574	138	176	34816	34816
3-Ports	270	286	107	126	26112	26112

Table 2. Resource utilization in Altera FPGA Stratix-II (EP2S180).

The power consumption for the proposed architecture is 1.73W as measured by Altera's Power Play tool with typical 12.5% toggle rate for four port architecture. In [19], 1.86W power consumption has been reported for their DSB-300 architecture by using the power models in Orion 2.0. Our four port architecture, used for simulation with 256-flit buffers with each flit of 34 bits is equivalent to DSB-1024, which is almost 3.5 times bigger than DSB-300. Thus, compared to [19], the proposed architecture is power efficient.

5 Discussion

The level of complexity is raising day by day for upcoming applications, which increases the design time proportionally. MPSoC is the solution to deal with this complexity. For MPSoC solutions, the biggest challenge is to deal with increasing gap between technological possibilities and design methodologies. NoC comes as a solution for the on-chip communication challenges of the future MPSoC architectures [25]. Many automated design flows for MPSoC and NoC platforms have been proposed with variety of features and solutions which can significantly reduce the design time as well [6, 7, 10]. But it is common thinking that when automated design methodology is used, the design time is reduced at the cost of performance.

In typical NoC mesh, processing nodes can be divided into three different categories according to the position : corner, edge and central nodes. Routers on the basis of these categories require three, four and five I/O ports respectively. Because of the distributed and independent control logic, router can be configured to fulfil these requirements without any extra effort by simply removing the extra hardware. Thus the architecture is fully optimized for any number of I/O ports. The number of I/O ports can be increased to any number by simply duplicating the hardware on input side and increasing the crossbar size on output side. At the same time, the buffer utilization is increased which significantly reduces the new buffer requirements.

The proposed architecture can be integrated into the existing automated NoC design flow without requiring extra effort. An automated design flow can utilize the adoptability feature of this architecture to generate an optimized router module according to the application and design requirements. Therefore, the proposed architecture can play a significant role in maintaining the performance of NoC based systems regardless of the topology.

6 Conclusions

In the current silicon era, NoC is not power and area efficient although it has higher throughput. Enhancement in the utilization of idle resources instead of inserting new ones can make the NoC an ideal solution for current applications. The designed prototype is based on parameterized and synthesized components which include FIFO's, input and output controllers, buffer allocator, output crossbar and route computation element. Due to the distributed nature of proposed architecture, increasing the number of ports is not an issue and hence it can be used in the implementation of a 3D NoC or long-range link insertions [18].

Future work. A real time application can be used to make the throughput analysis using different available routing techniques and can be compared with existing router architectures. Apart from that, different services like multicast and network monitoring can be introduced for performance improvement. A domain specific language for another distributed architecture has recently been introduced in [12]. We foresee that a similar approach can be taken for the further development of NoC systems, in order to raise the level of abstraction towards application layers. The router architectural features introduced here will be considered for an extended version of the mentioned DSL.

Acknowledgements. The present work is supported by the DOMES project funded by the Academy of Finland, project number 123518/2008.

References

[1] *Stratix II Device Handbook* 2007, Vol. 1 and 2, Altera.

[2] A. Jantsch and H. Tenhunen (Eds.). *Networks on Chip*, Kluwer Academic Publishers, 2003.

[3] A. Kodi, A. Louri, J. Wang. *Design of energy-efficient channel buffers with router bypassing for network-on-chips (NoCs).* Proceedings of International Symposium on Quality of Electronic Design (ISQED), pp.826-832, March 2009.

[4] T. Seceleanu. *The SegBus Platform - Architecture and Communication Mechanisms.* Journal of Systems Architecture (2006), doi:10.1016/j.sysarc.2006.07.002

[5] G. Luo-Feng et. all. *Design and performance evaluation of a 2D-mesh Network on Chip prototype using FPGA.* Proceedings of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp.1264-1267, 2008.

[6] D. Truscan et. all. *A Model-Based Design Process for the SegBus Distributed Architecture.* Proceedings of 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), pp. 307-316, 2008.

[7] D. Cozzi et. all. *Reconfigurable NoC design flow for multiple applications run-time mapping on FPGA devices.* Proceedings of the 19th ACM Great Lakes symposium on VLSI (GLSVLSI), pp.421-424, 2009.

[8] James Balfour and William J. Dally. *Design tradeoffs for tiled CMP on-chip networks.* Proceedings of the 20th annual international conference on Supercomputing (ICS), pp.187-198, 2006.

[9] K. Keutzer, A. R. Newton, J. M. Rabaey and A. Sangiovanni-Vincentelli. *System-Level Design: Orthogonalization of Concerns and Platform-Based Design.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 19, No. 12,December 2000, pp. 1523-1543.

[10] Khalid Latif, Moazzam Niazi, Hannu Tenhunen, Tiberiu Seceleanu, Sakir Sezer. *Application development flow for on-chip distributed architectures.* Proceedings of IEEE International SoC Conference (SOCC), Sept. 2008, pp. 163-168.

[11] Luca Benini and Giovanni De Micheli, *Networks on Chips.*, Morgan Kaufmann Publishers, 2006.

[12] Moazzam Niazi, Khalid Latif, Tiberiu Seceleanu, Hannu Tenhunen. *A DSL for the SegBus Platform.* Proceedings of IEEE International SoC Conference (SOCC), Sept. 2009, pp. 393-398.

[13] M. Coenen et. all. *A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control.* Proceedings of the 4th international conference on Hardware/software codesign and system synthesis (CODES+ISSS), pp.130-135, October 2006.

[14] M. H. Neishabouri, Zeljko Zilic. *Reliability aware NoC router architecture using input channel buffer sharing.* Proceedings of the 19th ACM Great Lakes symposium on VLSI (GLSVLSI), pp.511-516, 2009.

[15] N. Banerjee, P. Vellanki and K.S. Chatha. *A Power and Performance Model for Network-on-Chip Architectures.* Proceedings of the conference on Design, automation and test in Europe (DATE), pp.1250-1255, Vol.2, 2004.

[16] Robert Mullins, Andrew West and Simon Moore. *Low-Latency Virtual-Channel Routers for On-Chip Networks.* Proceedings of the 31st Annual IEEE International Symposium on Computer Architecture (ISCA), pp.188-197, 2004.

[17] T. T. Ye, L. Benini, G. De Micheli. *Analysis of power consumption on switch fabrics in network routers.* Proceedings of the 39th Design Automation Conference (DAC), pp. 524-529, 2002.

[18] Umit Y. Ogras, Radu Marculescu. *"It's a small world after all": NoC performance optimization via long-range link insertion.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, July 2006, pp. 693-706.

[19] V.Soteriou, R.S. Ramanujam, B. Lin, Li-Shiuan Peh. *A High-Throughput Distributed Shared-Buffer NoC Router.* IEEE Computer Architecture Letters, vol. 8, no. 1, pp. 21-24, Jan.-June 2009, doi:10.1109/L-CA.2009.5.

[20] W. Hangsheng, L. S. Peh, and S. Malik. *Power-driven design of router microarchitectures in on-chip networks.* Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 105-116, 2003.

[21] William James Dally, Brian Patrick Towles *Principles and Practices of Interconnection Networks* . The Morgan Kaufmann Series in Computer Architecture and Design, 2004.

[22] Xuning Chen and Li-Shiuan Peh. *Leakage power modeling and optimization of interconnection networks.* Proceedings of International Symposium on Low Power Electronics and Design, pp. 9095, 2003.

[23] Ying-Cherng Lan, Shih-Hsin Lo, Yueh-Chi Lin, Yu-Hen Hu, Sao-Jie Chen. *BiNoC: A bidirectional NoC architecture with dynamic self-reconfigurable channel.* Proceedings of 3rd ACM/IEEE International Symposium on Networks-on-Chip (NoCS), pp.266-275, May 2009.

[24] Tiberiu Seceleanu, Ville Leppänen, Olli S. Nevalainen. *Device allocation on the SegBus platform based on communication scheduling cost minimization.* Proceedings of the IEEE International SOC Conference (SOCC), Sept. 2007, pp. 191-196.

[25] Yassine Aydi, Samy Meftali, Mohamed Abid, Jean-Luc Dekeyser. *Dynamicity Analysis of Delta MINs for MPSOC Architectures.* In Conference internationale des sciences et = technique de l'automatique (ICM'07), Sousse, Tunisie, November 2007.