# An Area-efficient Dynamically Reconfigurable Spatial Division Multiplexing Network-on-Chip with Static Throughput Guarantee

Zhiyao Joseph Yang[1], Akash Kumar[1,2] and Yajun Ha[1]
[1]*Department of Electrical & Computer Engineering, National University of Singapore*
[2]*Eindhoven University of Technology, The Netherlands*
*Corresponding author email: akash@nus.edu.sg*

***Abstract*—With an increasing trend to implement Network-on-Chip (NoC)-based Multi-Processor Systems-on-Chips (MPSoCs), NoCs need to have guaranteed services and be dynamically reconfigurable. Many current NoCs consume too much area and cannot support dynamic reconfiguration. In this paper, we present an area-efficient Spatial Division Multiplexing (SDM)-based NoC. We replaced area consuming 32-bit to M-bit serializers with 32-bit to 1-bit serializers in the network interface and incur almost no loss in performance. We also restrict flexibility in the router to achieve further area reduction. A separate area-efficient control network, with an overhead of 3.9% of the total area of the NoC, is developed to support dynamic reconfiguration.**

***Index Terms*—FPGA, Network-on-Chip, throughput guarantee, dynamic reconfiguration, spatial division multiplexing.**

## I. Introduction

Chip density is increasing, allowing even larger systems to be implemented on a single chip. With increasing demands on flexibility and performance, these systems, known as Systems-on-Chips (SoCs), combine several types of processor cores, memories and custom modules of widely different sizes to form Multi-Processor Systems-on-Chips (MPSoCs). The bottleneck in such systems is shifting from computation to communication [1]. The traditional way of using bus-based mechanisms for inter-module communication has two main limitations. Firstly, it does not scale well with increasing system complexity. Secondly, it couples computation and communication of the system leading to longer design times. Networks-on-Chip (NoCs) have been proposed as an efficient and scalable alternative to shared buses which allow systems to be designed modularly.

Initial NoC proposals rely only on a Best-Effort service based on packet-based switching techniques. QNoC [2] and xPipes [3] are examples of packet-based NoCs. Another group of proposals is based on circuit-based techniques whose approach is to reserve the entire path from the source to the destination before data is sent out from the source. Examples include PNoC [4] and ProtoNoC [5]. For more demanding systems, it is necessary to have predictable performance as connections between the IP blocks are subjected to timing constraints. For such applications, it is necessary to be able to guarantee throughput before run-time. To achieve this, link allocation is done statically during design-time.

The most commonly used approach to guarantee throughput is Time Division Multiplexing (TDM) where different connections use the same links at different time slots. One downside of TDM is that the switching configuration of the router has to be updated every time slot. This requires slot tables that consume area and power in each of the routers. Examples of TDM-based NoCs are Æthereal [1] and Nostrum [6]. Spatial Division Multiplexing (SDM) [7] is another approach where subsets of the links which inter-connect the routers are allocated to different connections. Each connection thus has exclusive usage of the wires assigned to them. The sender serializes data on the wires allocated and the receiver deserializes the data before forwarding to the IP block. The advantage of SDM over TDM is that it eliminates the expensive slot tables thus reducing power but complexity is shifted to the serializers and deserializers.

From an application's perspective, applications having multiple use-cases require different sets of connection requirements at different times. A use-case is a combination of applications that run at the same time. Current NoCs cannot handle dynamism, therefore we need a new NoC which allows dynamic reconfiguration with different setups during run-time.

In this paper, we give solutions to the above problems, specifically the major contributions are as follows.

- A new design and detailed architecture for the Network Interface (NI) is presented to address the complexity of the serializers and deserializers that is inherent in SDM-based NoC. This results in significant area savings as the NI consumes a large proportion of area in the NoC.
- A low-complexity router design is presented which results in lower area usage at the cost of routability. This also eliminates the problem of reordering the data when they arrive at the receiving NI.
- The functionality of dynamically reconfiguring the NoC with different connection setups during run-time.
- The development of a design tool which generates VHDL files for a NoC based on user-specified parameters. The tool is available online [8] for use by the community.
- An algorithm which finds link allocations needed to satisfy a given set of connection requirements for varying router flexibility.

Point-to-point links between MicroBlazes in a Multi-Processor System-on-Chip (MPSoC) generated using [9] have been replaced by a NoC generated with our tool and a working prototype has been successfully implemented on a Xilinx FPGA development board (XUPV2P). The MicroBlazes were

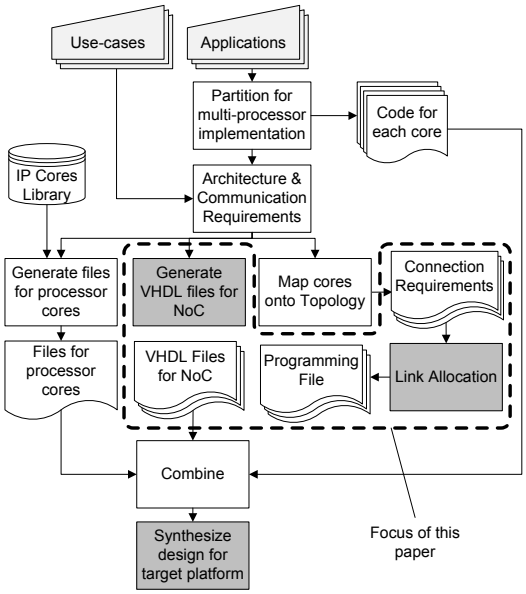



Fig. 1. Design Flow

used to verify the dynamic reconfiguration of the NoC as well as to send and receive data between each other.

The remainder of this paper is organized as follows. Section II shows the steps involved from application specifications to synthesizing the design for the target platform. Section III describes the architecture of the NoC developed and also design details of the NI and the router. Section IV evaluates experimental results and Section V concludes the paper and highlights future work.

## II. Design Flow

Figure 1 shows the design flow of an application to be implemented as an MPSoC on a target platform. The application is first partitioned for a multi-processor implementation which involves deciding how many cores to use, which core to perform which functions, etc. Examples of such tools are discussed in [9], [11] and [12]. After this stage, codes for each processor as well as the architecture and communication requirements for the NoC are obtained. With the architecture and communication requirements, files for the processor cores can be generated and the cores can be mapped to the topology chosen. VHDL files for the NoC will also be generated at this stage. This paper focuses on stages within the dashed region in Figure 1.

### A. Dynamic Reconfigurability

Figure 2 shows an example of three applications mapped onto a 2-by-2 NoC with one sending channel and one receiving channel per node. The dashed, dotted and solid arrows represent the connection requirements for the three applications. We see that we cannot satisfy the requirements as two sending/receiving channels are required at some nodes when only one is available. However, if it is specified that there are only two use-cases (as shown in Figure 2), then we can perform the link allocation for two setups (one for each use-case) and reconfigure the NoC for the appropriate use-case when needed during run-time.

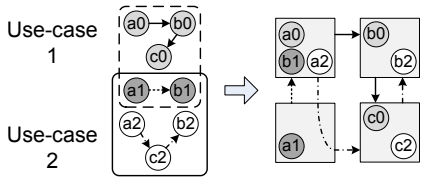



Fig. 2. Two Use-cases with Three Applications

### B. Link Allocation

Performing link allocation at design-time ensures guaranteed performance right from the start. Requesting and allocating resources during run-time runs the risk of not having the available resources and thus not being able to satisfy requirements. We employ a recursive mapping function which utilizes backtracking to find the configuration that satisfies all connection requirements. Although this will result in a more complex path finding algorithm, this is acceptable as it is done at design-time.

## III. Network Architecture

The network architecture is designed as a two-layer system where one layer is data network responsible for transporting data while the other layer is the control network responsible for programming the NoC. Each node in the control network will provide the link configurations to the corresponding node in the data network.

### A. Control and Data Network

The control network is a low cost network as it is only used when the NoC needs to be programmed. For our test setup, the control network has an overhead of 3.9% of the total area of the NoC. The data which is used to program the NoC is transmitted to the control network via a master node.

Nodes in the control network are interconnected using the minimum number of links required. The only requirement is that there is a path from the master node to all other nodes. Thus we can sacrifice path diversity and return paths in favour of lower area usage. The bytes are processed and forwarded based on the protocol developed. The number of programming bytes needed is:

$$NumOfBytes = 2 + [12 + 3(NumOfSendingChannels + NumOfReceivingChannels)](NumOfNodes)$$

For the *data-network*, the nodes as well as the NI are interconnected via two links (one in each direction). The number of wires in each link is user-defined as a parameter in the design tool. Each node in the data network has a NI which will interface to an IP core. The NI does the necessary serialization and deserialization as data pass from the IP core to the router and vice versa.

### B. Network Interface

In [7], the NI design for the SDM approach presented is highly flexible but incurs high area cost. Figure 3 shows the NI design as interpreted from [7] for the case of eight wires per port, two sending channels and 32-bit data width. For each sending channel, an output message queue buffers 32-bit data from the IP core. The data is then sent to a 32-bit

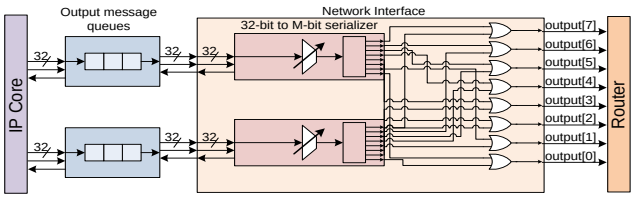



Fig. 3. NI Design [7]

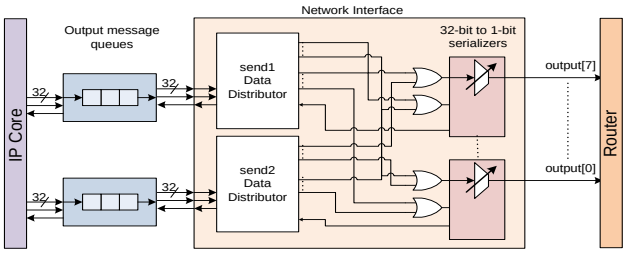



Fig. 4. New NI Design

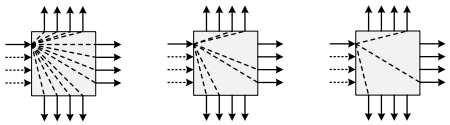
to M-bit serializer which serializes the data onto the allocated wires. The 32-bit to M-bit serializer is flexible and the value of M depends on how many wires have been allocated to that particular sending channel. Our design achieves lower complexity and area savings by using 32-bit to 1-bit serializers instead of 32-bit to M-bit serializers.

Figure 4 shows the new NI design developed for the same case. Instead of 32-bit to M-bit serializers (Figure 3), there are multiple 32-bit to 1-bit serializers. The number of serializers will depend on how many wires there are per port (eight for the case in Figure 4). In this design, there is one data distributor for each sending channel. The data distributor is responsible for forwarding the 32-bit data to the serializer that has been allocated to that particular sending channel. As input to the 32-bit to 1-bit serializer, there is one OR gate to allow all the data distributors to be able to forward data to each serializer. The other OR gate and 1-bit output are hand-shaking signals between the data distributor and the serializer.

The advantage of the new NI design is area savings. The trade off is that latency is affected when not sending data continuously. For example, it will take longer to send one 32-bit data using the new NI design compared to the NI design in [7]. However, for the case of sending a stream of data continuously, latency is comparable for both NI designs.

### C. $k$-way Router

For the router described in [7], there is full flexibility which means each incoming link is allowed to use any of the outgoing links. This results in a big switch required in each router. We use the term *$k$-way router* to describe the degree of flexibility of the router where $k$ refers to the number of options available at each outgoing port for an incoming link. Figure 5(a) shows an example of a 4-way router where the first incoming link from the west can use any of the four wires of the outgoing links in each direction. Besides the high area cost incurred for routers with full flexibility, another issue is that the order of the data at the sending NI may not be the same when the data reaches the receiving NI. In [10], the idea of reducing flexibility in the FPGA switch blocks is discussed.



(a) 4-way (b) 2-way (c) 1-way

Fig. 5. $k$-way Router

TABLE I
COMPARISON OF SWITCH SIZE

| Router | Evolution of Number of Cross Points |
|---|---|
| Full Flexibility | $O(N^2)$ |
| Benes Switch [7] | $O(N \cdot log_2(N))$ |
| Our Design | $O(N)$ |

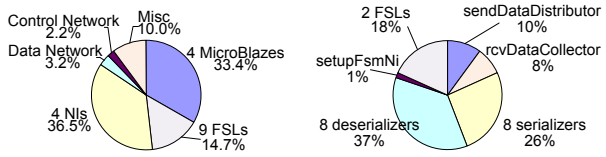
The case of a FPGA switch block is similar to the $k$-way router as the objective for both cases is to allocate links based on given requirements at design-time. However, to the best of our knowledge, this idea has not been used in the NoC context. Figure 5 shows the options available to an incoming wire for $k$-way routers with different values of $k$. By restricting the number of links that each incoming wire can use, the area cost of the router is reduced. The 1-way router seems to be a good choice as it solves the problem of ordering of the data at the receiving NI while using significantly less area. Table I shows a comparison of the switch size in terms of the number of cross points required. A cross point is a small switching element that makes or breaks a connection between one input wire and one output wire of the switch.

## IV. EXPERIMENTS AND RESULTS

In order to verify the functionality of the NoC, a system with 4 MicroBlazes was generated using [9] in which point-to-point links were replaced by a NoC generated with our tool. The four MicroBlazes are connected to the NoC via Fast Simplex Links (FSLs) and were used to verify the dynamic reconfiguration of the NoC as well as to send and receive data between each other. A MicroBlaze sends programming data via the additional FSL to program the NoC. A laptop connected via UART to that MicroBlaze allows debugging data to be sent/received. The design has successfully been implemented on a Xilinx FPGA development board (XUPV2P) and used to study the area, power distribution and scalability on FPGA.

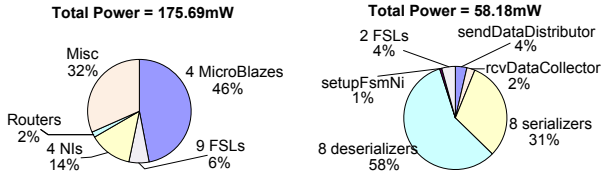
### A. Area and Power on FPGA

Figure 6 and 7 shows the area and power (obtained using Xilinx XPower Analyzer) breakdown of the whole design and the NI with one sending, one receiving channels and eight wires per input/output port. Each FSL buffers 64 words of 32-bits each. The MicroBlazes are implemented very efficiently by Xilinx. Optimizing the NI brings about great savings to the design as a whole as it takes up a significant proportion of the total area. The serializers and deserializers use more area than the sendDataDistributor and rcvDataCollector in this example as there is only one sending and one receiving channel. The number of components required in the NI is discussed in Section III-B. The serializers and deserializers consume more power than the other NI components as they are more active components. The routers consume minimal power




(a) Top (b) NI

Fig. 6. Area Breakdown




(a) Top (b) NI

Fig. 7. Power Breakdown

TABLE II
COMPARISON OF NI COMPONENTS

| Component | Number of Slices | Power (mW) |
|---|---|---|
| 32-bit to M-bit serializer [7] | 13319 | 134.38 |
| sendDataDistributor | 183 | 2.08 |
| 32-bit to 1-bit serializer | 48 | 2.27 |

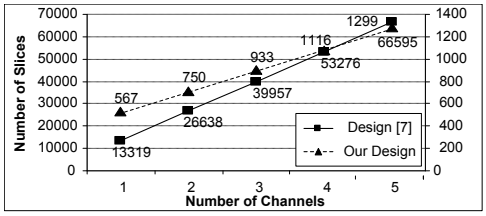



Fig. 8. Device Utilization Comparison of NI Design with Varying Number of Channels

due to minimal switching required in a SDM-based design. The control network, which supports dynamic reconfiguration, has an overhead of 3.9% of the total area of the NoC.

### B. Area Scalability

Our new design is more scalable mainly due to the reduced complexity in the NIs and routers.

*1) Network Interface:* Table II shows a comparison between the main components used in both NI designs. Our design consumes less area and power. Figure 8 shows the device utilization for NIs having eight wires per port with varying number of channels required (note the different vertical scales used). We have achieved area savings of over 95% with almost no loss in performance.

*2) Router:* Figure 9 shows the number of slices used for the $k$-way router with eight wires per link in each direction as we decrease the flexibility. The area cost decreases as we decrease the value of $k$. The 1-way router is desirable as it eliminates the need to reorder data at the receiving NI and lowers total area cost. However, the trade-off is that we are not able to satisfy all cases that we can with full flexibility but we have highlighted ways to avoid such situations (Section III-C).

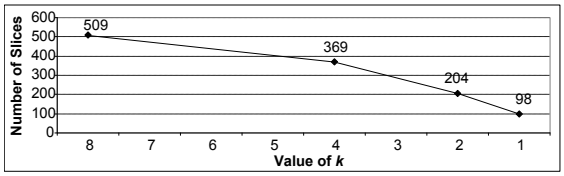



Fig. 9. Device Utilization of $k$-way Router

## V. CONCLUSION AND FUTURE WORK

In this paper, we have presented new designs and detailed architecture for the NI and router which achieved area savings of more than 95% and improved scalability. Performing link allocation during design-time allows us to provide throughput guarantees as well as to generate different programming files for different use-cases which are used to dynamically configure the NoC during run-time.

In the future, the feasibility of connection requests from IP cores during run-time will be explored. Another aspect to be explored is flow control. Further, we plan to evaluate the trade-offs of the 1-way router in the future.

## ACKNOWLEDGMENT

The authors would like to thank Mr Shakith Fernando for his assistance in the development of the MPSoC platform.

## REFERENCES

[1] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[2] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, no. 2-3, pp. 105–128, 2004.

[3] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

[4] C. Hilton and B. Nelson, "PNoC: A flexible circuit-switched NoC for FPGA-based systems," *IEE Proceedings: Computers and Digital Techniques*, vol. 153, no. 3, pp. 181–188, 2006.

[5] D. Castells-Rufas, J. Joven, and J. Carrabina, "A validation and performance evaluation tool for ProtoNoC," *2006 International Symposium on System-on-Chip, SOC*, 2006.

[6] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," *Proceedings - Design, Automation and Test in Europe Conference and Exhibition*, vol. 2, pp. 890–895, 2004.

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

[8] NoC Generation Tool. `Username: fpl10 Password: fpl10`. [Online]. Available: http://www.ics.ele.tue.nl/~akash/NoCGenTool/

[9] A. Kumar, S. Fernando, Y. Ha, B. Mesman, and H. Corporaal, "Multiprocessor System-level Synthesis for Multiple Applications on Platform FPGA," in *Proceedings of 17th International Conference on Field Programmable Logic and Applications*. IEEE Circuits and Systems Society, 2007, pp. 92–97.

[10] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 277–282, 1991.

[11] B. De Sutter, D. Verkest, E. Brockmeyer, E. Delfosse, A. Vandecappelle, and J. Mignolet, "Design and tool flow of multimedia MPSoC platforms," *Journal of Signal Processing Systems*, vol. 57, no. 2, 2009.

[12] S. Kwon, C. Lee, and S. Ha, "Data-parallel code generation from synchronous dataflow specification of multimedia applications," *Proceedings of the 2007 IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia, ESTIMedia 2007*, pp. 91–96, 2007.