Reconfigurability of On-chip Networks

Muhammad Obaidullah*, Abdullah Siddiqui†, Adeem Mustafa‡, Dr. Lev Kirischian§

Electrical & Computer Engineering Department, Ryerson University,
Toronto, M5B 2K3, Canada

*mobaidullah@ryerson.ca

†abdullah.siddiqui@ryerson.ca

‡adeem.mustafa@gmail.com

§lkirisch@ee.ryerson.ca

Abstract

Nowadays, chips are not designed to serve just one purpose (application mode), rather they are designed to adapt and reconfigure to different constraints. This reconfigurability and adaptability can be enhanced by using a interconnection network which in of itself is reconfigurable. This research paper aims to address most reconfigurable aspects of on-chip interconnection networks including reconfigurable synchronous/asynchronous links, link width, virtual channels, switching techniques, network topology, buffer depth, and router pipeline. Static, dynamic, and mixed network reconfigurability is also discussed in detail. Additionally, concept of virtual hardware components with loading controller and self-loading virtual hardware components is also discussed along with similarities and differences.

Keywords

Fabric, Interconnection, NoC (Network-on-Chip), Capacity, Latency, FLITS (FLow Control UnITS), VHC (Virtual Hardware Component)

I. Introduction

For many decades people have speculated that Moore's Law [1] might not be true in the next few years, but every time technological advancement has surprised us all. The number of cores on a single die are increasing as a result of advancement in nanometer technology. 14nm chips were already developed by Samsung in December 2012 and research is going on for breaking the 10nm barrier.

Cramming more and more transistors onto single die has lead the IC designers to integrate not one but many modules and cores onto single chip and inter connecting them. Like a computer needs to talk to a peripheral device for input, output, or signal processing, similarly cores and modules inside the chip need to talk to each other for passing information.

Conventionally, modules were directly connected using buses but as number of modules increased, the complexity of the chip grew. To deal with complexity, abstraction and regularity of design is required.[2]

IC designers began to think of a network to topology to connect these cores and modules. [3] Common network topologies to interconnect cores include bus, ring, two-dimensional mesh, and crossbar.

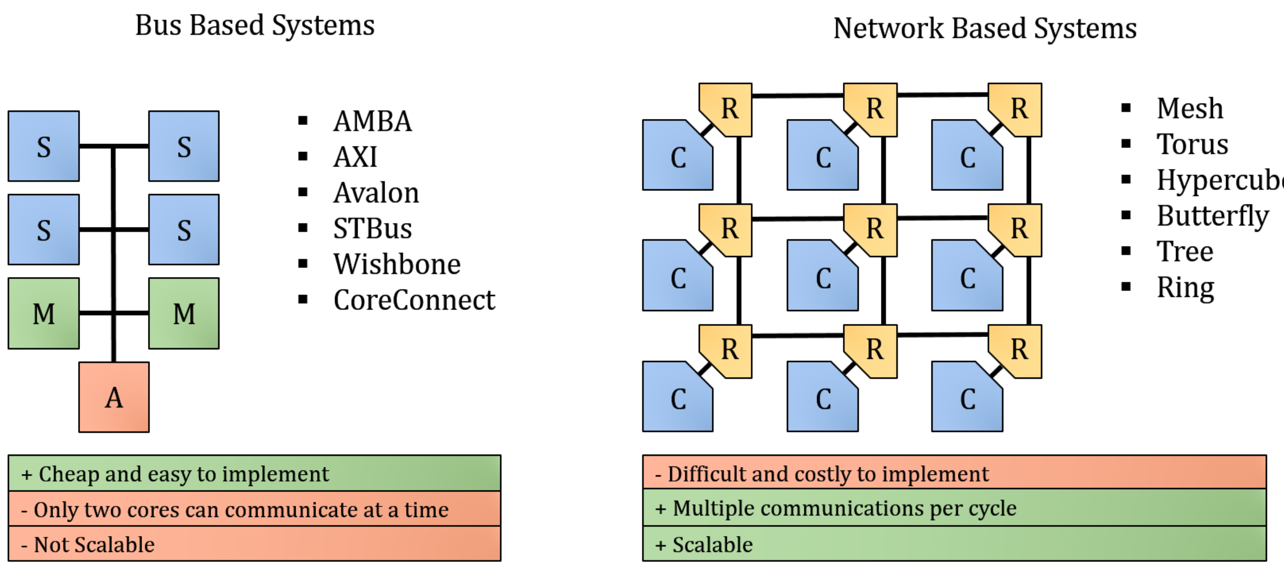


Figure 1: Differences in bus-based and network-based systems.

Traditionally, IPs or cores were mapped using static bus architectures eg. AMBA, AXI, STBus etc. But the bus architecture has many drawbacks when increasing the number of IP blocks on the SoC. Every IP block attached in bus adds a parasitic capacitance degrading the signal integrity. Bandwidth is limited and is shared among all IPs attached. Only one transaction can occur in a single clock cycle. Bus arbitration logic delay grows as more number of masters are added onto bus.[4]

In order to achieve maximum performance from the hardware, the interconnection medium should change/reconfigure to meet current application's demands. There several ways this can be done and are discussed in this paper.

II. Reconfigurability of Network On Chip

There are many parameters or features that of an on-chip network that can be changed or reconfigured in order to meet specifications (combination of demands/constraints/costs/application modes)

1) **Synchronization of Links:** The links of the network can be individually (locally) or as a whole (globally) changed to transport asynchronous or synchronous signals. (*GALS* - Globally Asynchronous Locally Synchronous concept)
2) **Link Width:** The flits can be broken down into more smaller units and sent through the link if multiple flits needs to travel through the same link at the same clock cycle. This increases number

of flits allowed to traverse the switch at a time but increases latency.

3) **Virtual Channels:** Use of virtual channels allow multiple packets to traverse through the switch by sharing the link in time. (Time Division Multiplexing - TDM)

4) **Switching Techniques:** There are three major switching techniques used in NoC (Network On Chip). These techniques can be deployed on link-level, router-level, or network-level. If faster data transfer rate is required without interruption, circuit switching technique is preferred. If multiple data transfers are required and the data is not time-sensitive, packet switching or virtual circuit switching technique can be deployed.

5) **Network Topology:** Based on the current application mode or environmental conditions, a router can disappear and become direct link between other routers. This reduces hop count and latency of a packet. The other approach to meet specifications is to change the topology itself by placing the routers in a switching fabric of links and loading the configuration bit file on the link joints to connect the router together. This concept is similar to the FPGA but in place of LUTs (Look Up Tables), the connection fabric contains routers.

6) **Buffer Depth:** The depth of buffers in a router can be modified to meet latency and area requirements of a chip. Hardware such ViChar Dynamic Virtual Channel buffers allow banks of buffers to be used and shared among all input ports of router which allows efficient use of chip area.

7) **Router Pipeline:** Reconfiguring the amount of processing done on a packet on arrival at input port, can drastically change the latency of the packet. Effective use of router pipeline and employing parallel processing reduce the packet latency but requires additional hardware and as a result more chip area.

A. *Types of Network Reconfigurability*

1) ***Static Network Reconfiguration (Offline):*** This type of reconfigurable system reconfigures itself before going into any application mode. In other words, when no application is currently executing, the network configuration for the next application is loaded and the network is reconfigured. Configuring the network statically improves utilization of space and reduces latency but at a small cost of reconfiguration time.

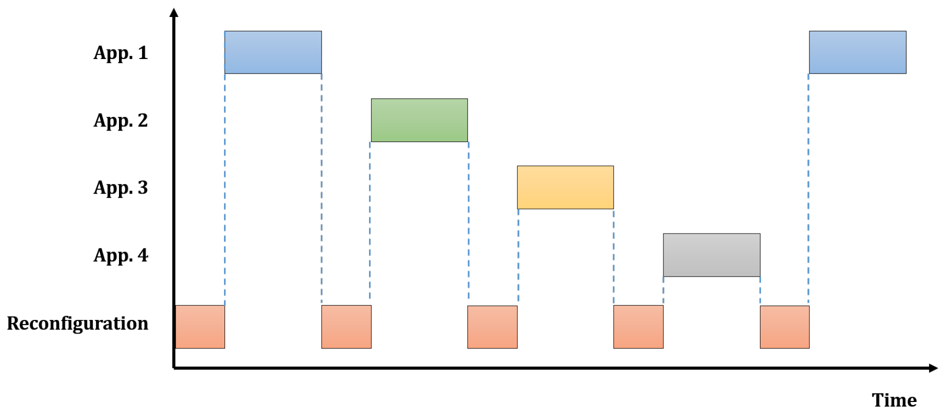


Figure 2: Static reconfiguration requires additional time between two application modes.

2) ***Dynamic Network Reconfiguration (Online):*** This type of reconfiguration allows the network bend to current network traffic. The amount and type of traffic depends on the current executing application. The network may choose to modify its architecture or routing methodology in order to accommodate rise or fall in traffic volume. This type of reconfiguration happens during the execution of the application. Reconfiguring dynamically still requires some kind of synchronization of data so that the packets/flits come in the required order.

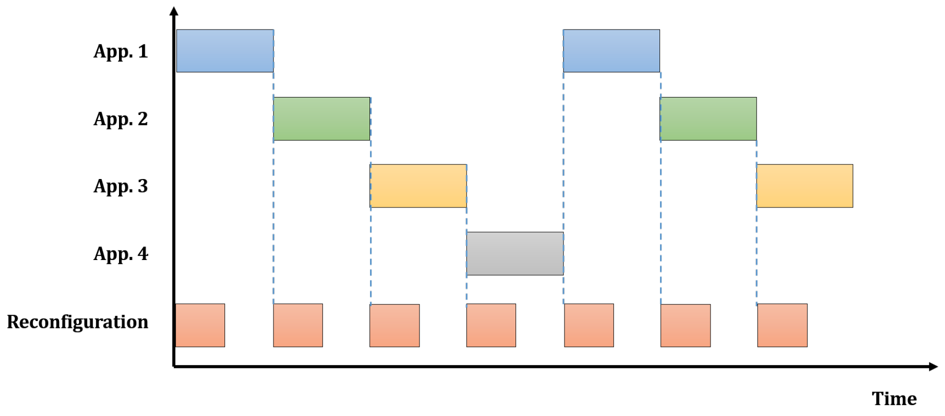


Figure 3: Dynamic reconfiguration happens during application execution.

3) ***Mixed Network Reconfiguration (Online & Offline):*** In this type of reconfiguration, some part of system is configured before the application starts executing and the remaining configuration is done during application execution. Usually, architectural changes such as Network Topology, Link Width, and Synchronous or Asynchronous links are reconfigured statically while parametric changes such as

virtual channels, buffer depth, ad router pipeline are reconfigured dynamically. Although one might argue that virtual channels are architectural components, it all depends on the implementation of component.

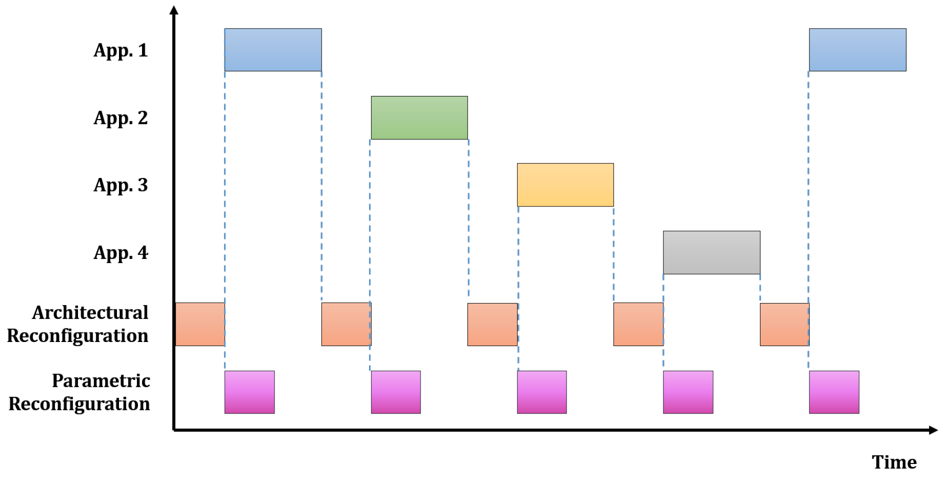


Figure 4: Parametric reconfiguration happens during application execution according to application needs while architectural reconfiguration occurs before application execution.

III. Literature Review

A. *Synchronous or Asynchronous*

When an electronic device transmits a digital or analog signal to the other electronic device, there is a certain rhythm established between these two devices, that is synchronization.

Synchronization respects other device's processing speed, task schedule, design constraints and conforms to a certain communication protocol. It is because of these properties of synchronous communication, it gets bad publicity. When system designers are looking for speed and performance rather than quality of service, synchronized communication does not qualify as a suitable candidate.

Each network node connects to a network interface which communicates with them synchronously and provides decoding and encoding features to send messages across the network.

This kind of network reconfigurability allows speedup of data transfer compared to worst-case scenario delay assumptions taken during pre-run period. However, data coherency and frequency sweep additional hardware is required which can add to chip area.

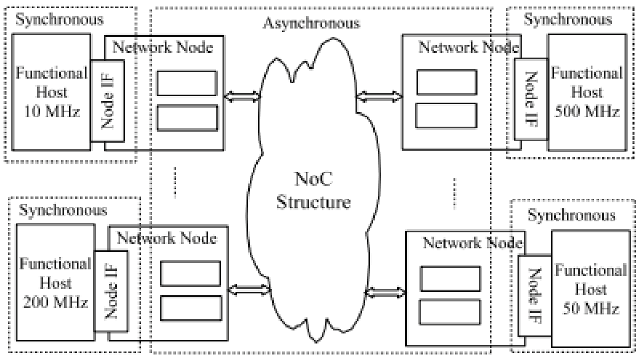


Figure 5: Applying GALS scheme in a NoC design.

B. Link Width

When there is congestion due to high communication volume in the network, need might arise to increase the throughput of existing links. Links can be split or shared between many destination or sources to get the packet traverse across the router.

Links can be shared for allowing different type of switching. The points on NoC where latency and contention is an issue, the links can be shared by two different switching mechanisms at the same time. On the sender side, the flits are divided into two equal parts and sent during two clock cycles. This allows data which is urgent to pass through the circuit switched part simultaneously.

Links can also be shared for spatially biased traffic congestion where one side input/output port of the router has more traffic than the other ports. Daihan Wang *et. al.* [5] propose to dynamically reconfigure the the link-width of the on-chip router in order to balance the load and optimize to spatially biased traffic. In this type of reconfiguration, there are three major steps:

1) An unbalanced router was designed with fine-grained centralized channel buffer shared by each port. Adjusting the link-width, the packet format is automatically updated.
2) The buffer data-width configuration data is kept in tables to keep track of currently allocated data-width and update it according to the current traffic conditions.

C. Virtual Channels

Demonstrations have shown that buffers are responsible for the consumption of the largest fraction of dynamic and leakage power of the NoC node. Reducing the number of buffers and increasing their utilization will reduce their power consumption and the area occupied by them. A number of intermediate switches can be occupied by a packet at the same time. To avoid this scenario, virtual channels are used. Using virtual

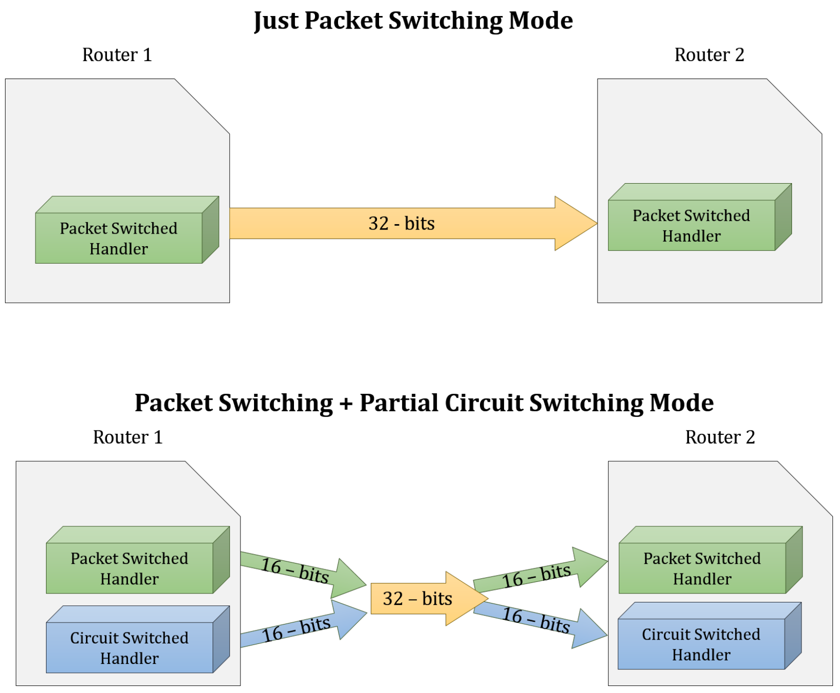


Figure 6: If the dual operation mode is required, the link width n can be divided into two equal parts $n/2$ so that the Packet flits can travel at the same time as circuit switched ones.

channels improves the throughput and allows blocked packets to be bypassed. This is a very interesting type of reconfiguration of network where the network automatically adjusts the hardware to conform to current traffic situation.

Latif et al. proposed a novel architecture for NoC routers which involves sharing of the VC buffers. The overhead is minimal and the utilization of resources, especially in the presence of faults is improved. The required performance is hence retained. This was made possible by reconfiguring the utilization of resources that cannot be accessed due to faults. As a result of sharing, this approach increases VC use because free buffers can be used by other channels. Optimal utilization of VC can be achieved by sharing among all input ports. This increases control logic complexity and power consumption. A tradeoff between power consumption and resource utilization can be achieved by forming groups with limited numbers of input ports which share resources according to communication requirements [6].

The architecture was proved to be tolerant of routing logic faults without the need of extra logic. The architecture was simulated with uniform, transpose, and negative exponential distribution synthetic traffic. It was also simulated with video conference encoder application traffic. Experiments compared the proposed architecture (PVS- NoC) with the FVS-NoC architecture and the symmetric 3-NoC architecture. The results showed that the proposed PVS-NoC architecture provides the best tradeoff between APL, power consumption

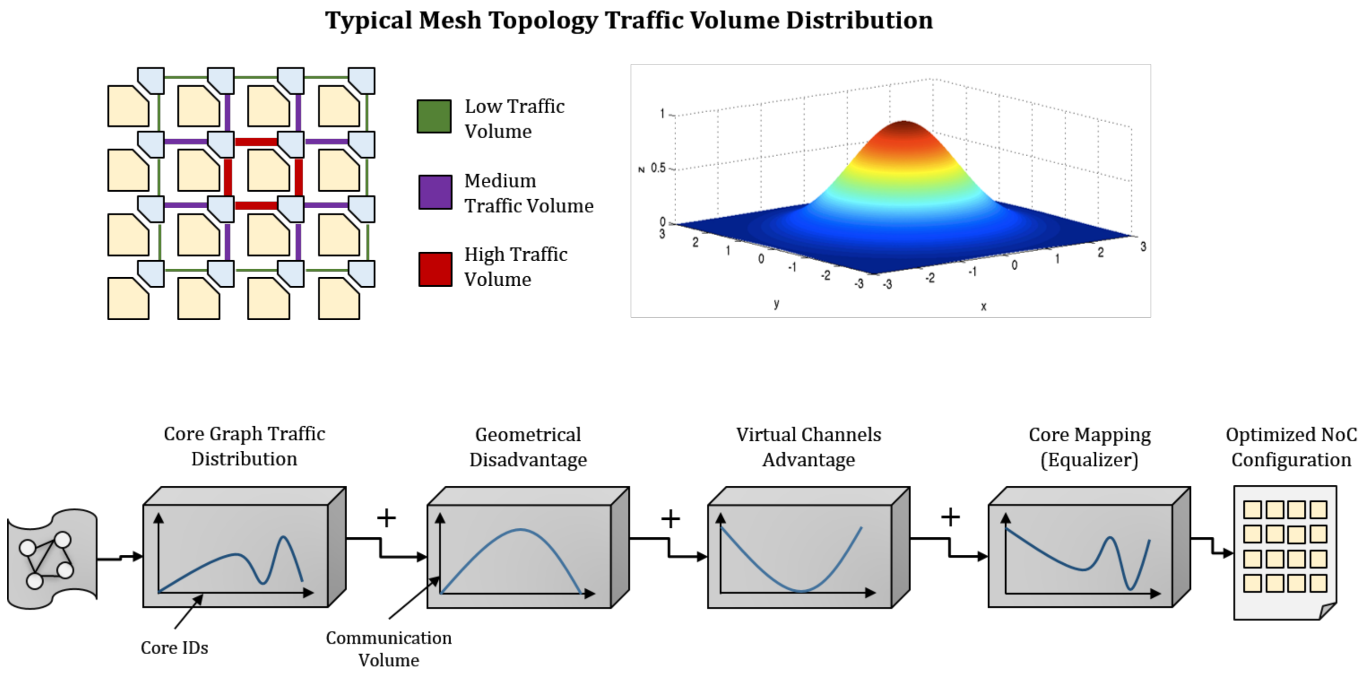


Figure 7: Optimal utilization of virtual channel buffers can distribute the traffic load equally in a network and can overcome some topology's geometrical disadvantages.

and silicon area. For the video conference encoder application, the PVS architecture exhibited considerable reduction in average packet latency and a minor overhead of silicon area and power consumption.Hence, it was concluded that PVS offers a superior trade-off in providing almost maximum performance at nearly minimum area and power costs [6].

Dynamically allocated multiqueues (DAMQs) are effective for achieving VC flow control with maximum buffer utilization. However, DAMQs have a number of limitations such as complex hardware, a queue structure which is suitable only for deterministic routing, interventions among VCs which cause high traffic congestions and large delays with flit arrivals/departures. In efficient dynamic VC (EDVC) approach, the model proposed by Oveis-Gharan and Khan, VCs employ variable number of buffer slots depending on the traffic. A state-of-the-art microarchitecture of input ports was proposed for flexible VC organization [7].

The EDVC mechanism employs logic circuits instead of tables for managing VC slots and saves one clock cycle for each flit arrival/departure from the input port, uses a simpler congestion mechanism, has no configuration constraints and is considerably simpler than table-based mechanisms (it mainly has pointers which have a scalable structure for optimizing performance) [7].

The hardware requirement and performance of the EDVC mechanism was compared with link-list CDVC and ViCHaR mechanisms. Results showed that an EDVC input port consumes 10% fewer registers for

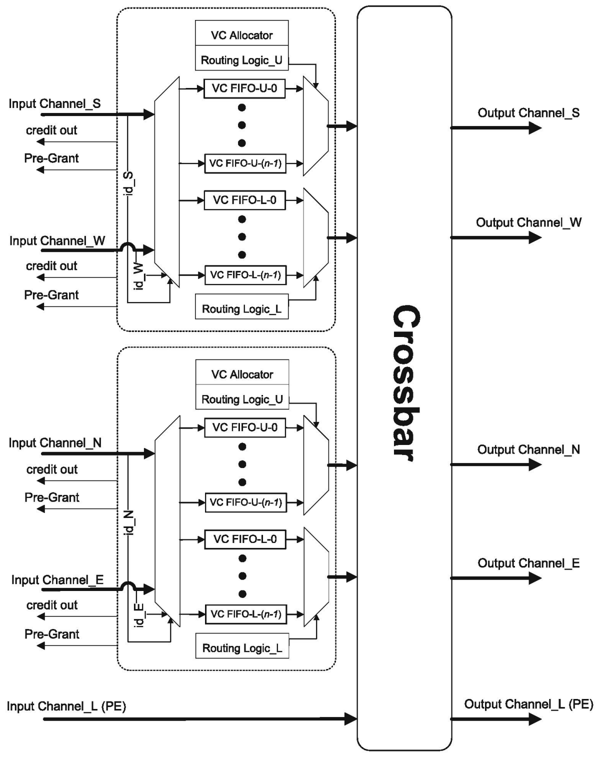


Figure 8: PVS approach for VC architecture. Image Source: [6]

FPGA implementation and 61% lesser power for ASIC design. It exhibited better latency (by 48-50%) and throughput (by 100%) in comparison with the CDVC approach for application-specific traffic in the NoC system. It also showed better performance for hotspot, tornado and complement traffic patterns when compared with CDVC and ViChar techniques [7].

In the NoC domain, HoL blocking increases latency and leads to lower throughput and buffer utilization. This problem can be eliminated by employing virtual channels. They enable multiplexing and buffering of packers in a single router channel. However, it does not remove HoL blocking completely. The methodology proposed by Oveis-Gharan and Khan, namely the PBVC approach, provides an effective solution to remove HoL blocking [8].

In the PBVC approach, the probability of getting a free VC for unblocked packets of a channel is greater than the CWVC mechanism. When a packet is blocked, its VC gets minimum space from the channel buffer. New packets stay in the upstream routers until a VC becomes empty. HoL blocking is avoided and each packet of a series reaches the destination in order. The PBVC mechanism also exhibits better buffer utilization when there is a large number of HoL blocking [8].

The only difference between the conventional wormhole VCs(CWVC) and PBVC NoC router organizations and architectures is the organization of their input-ports. A little extra hardware is needed to implement the PBVC approach on each router [8].

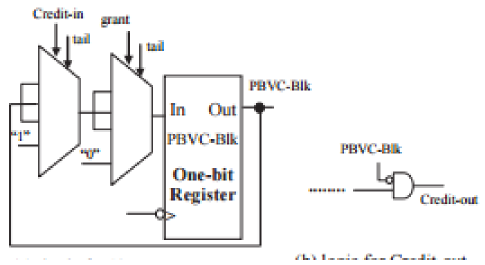


Figure 9: Extra hardware needed per VC for PBVC router input-port. Image Source:[8]

PBVC and CWVC were implemented using DAMQ buffers. Random and HoL specific traffic patterns to PBVC and CWVC based NoCs. Throughput and latency were also compared for both PBVC and CWVC based NoCs. Results showed that PBVC results are on average better than CWVC [8].

Interconnection technology is a limiting factor in future SoC designs. A possible solution is on-cip interconnection network. It provides an on-chip communication infrastructure for interconnecting system components [9].

A packet switching network with virtual channel flow control is proposed. The router architecture simplifies dynamic arbitration [9].

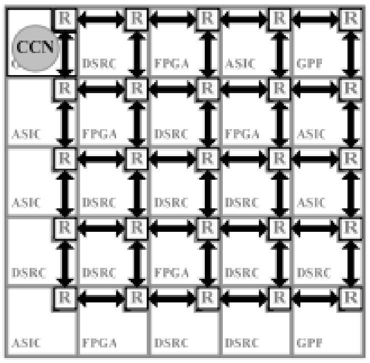


Figure 10: SoC on which the proposed architecture will be used. Image Source:[9]

Arbiter has to be deterministic and fair. The proposed architecture is smaller in size, has fair deterministic arbitration and high throughput [9].

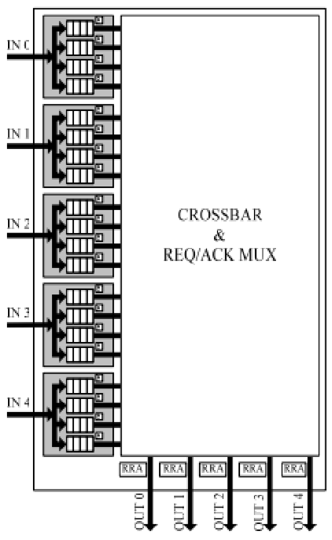


Figure 11: Proposed architecture. Image Source:[9]

A comparison was done between the traditional architecture (and SLIP arbiter) and the proposed architecture. The size of the proposed router was found to be reduced by 49% for the FPGA implementation and 23% for the ASIC implementation while being 1.4 times faster [9].

There is a general trend in recent works that whenever reconfigurability is required, a list or table memory is used to store some configuration bits and reconfigure the hardware for optimal performance.

D. Switching Techniques

Guoyue Jiang *et. al.* [10] propose a hybrid router capable of handling different switching techniques at the same time.

They use the application core graph to determine off-line the best switching technique to adopt on a link. Each link can be circuit switched (β), packet switched (α), or virtual circuit switched (γ).

Since branch-and-bound algorithm is of deterministic nature, as the solution space expands, the time to reach optimal solution will grow exponentially. Additionally, the paper concludes that the mapping technique consumes more time compared to NMAP, NormalBB, and (Discrete Particle Swarm Optimization) DPSO because the mapping is implemented statically.

Furthermore, since the algorithm looks for three different possibilities of switching (α, β, γ) in each link, it expands the search space by $3! = 3 \times 2 \times 1 = 6$ times. Although the search tree stops further branching out if any premature solution does not satisfy any one of the branching conditions, the time to solutions is still huge.

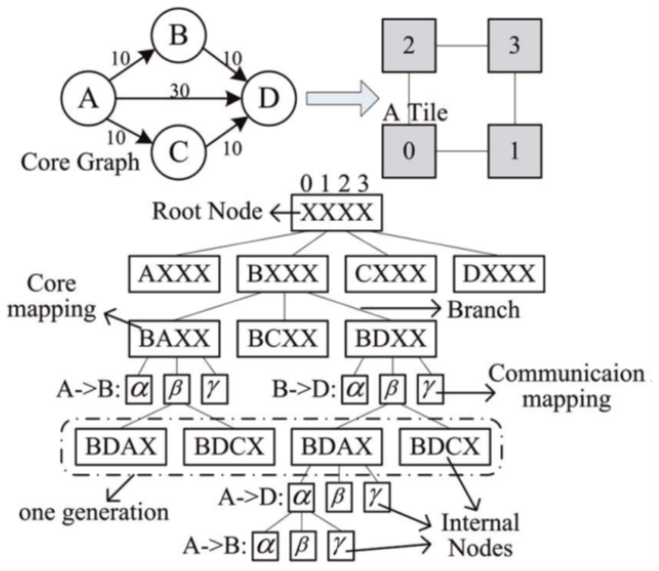
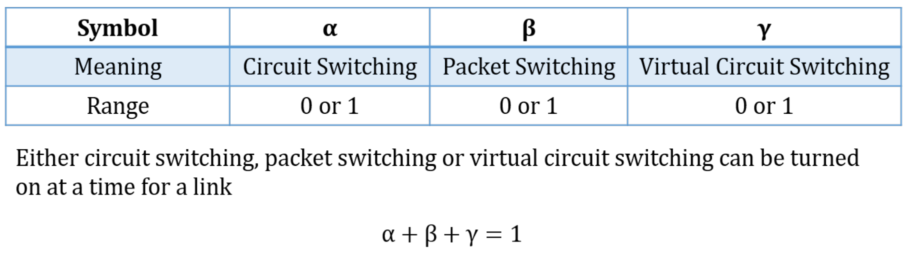


Figure 12: Figure shows how a decision is taken to make a communication link circuit switched, packet switched, or virtual circuit switched.



Symbol	α	β	γ
Meaning	Circuit Switching	Packet Switching	Virtual Circuit Switching
Range	0 or 1	0 or 1	0 or 1

Either circuit switching, packet switching or virtual circuit switching can be turned on at a time for a link

$$\alpha + \beta + \gamma = 1$$

Figure 13: For an edge, either circuit switching, packet switching, or virtual circuit switching can be turned ON.

E. Network Topology

In real world applications, there are many modes of operation (also known as application modes) where the traffic conditions drastically change. This can cause humongous loss of power and increase latency if the network also does not conform to changes in traffic. When a different mode of operation is selected, the cores must be remapped to optimal positions where there is minimum energy consumption and latency.

The question then arises, how can cores be remapped and connect differently according to application mode. For this, Paria & Hamid [11] propose to reduce the hop count between cores (IP blocks) which require high communication volume using combination of crossbar switches and routers. The authors assume a typical mesh topology with each router having five links (North, South, East, West and Local). Each router

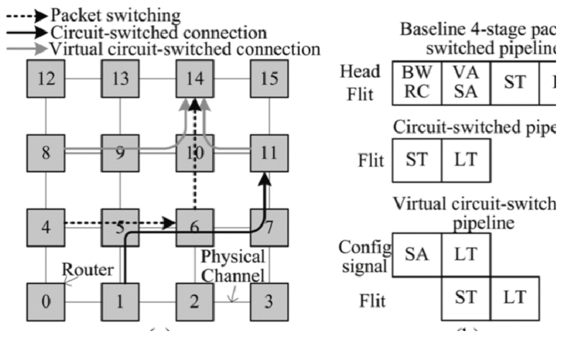


Figure 14: Proposed network architecture with hybrid communication links.

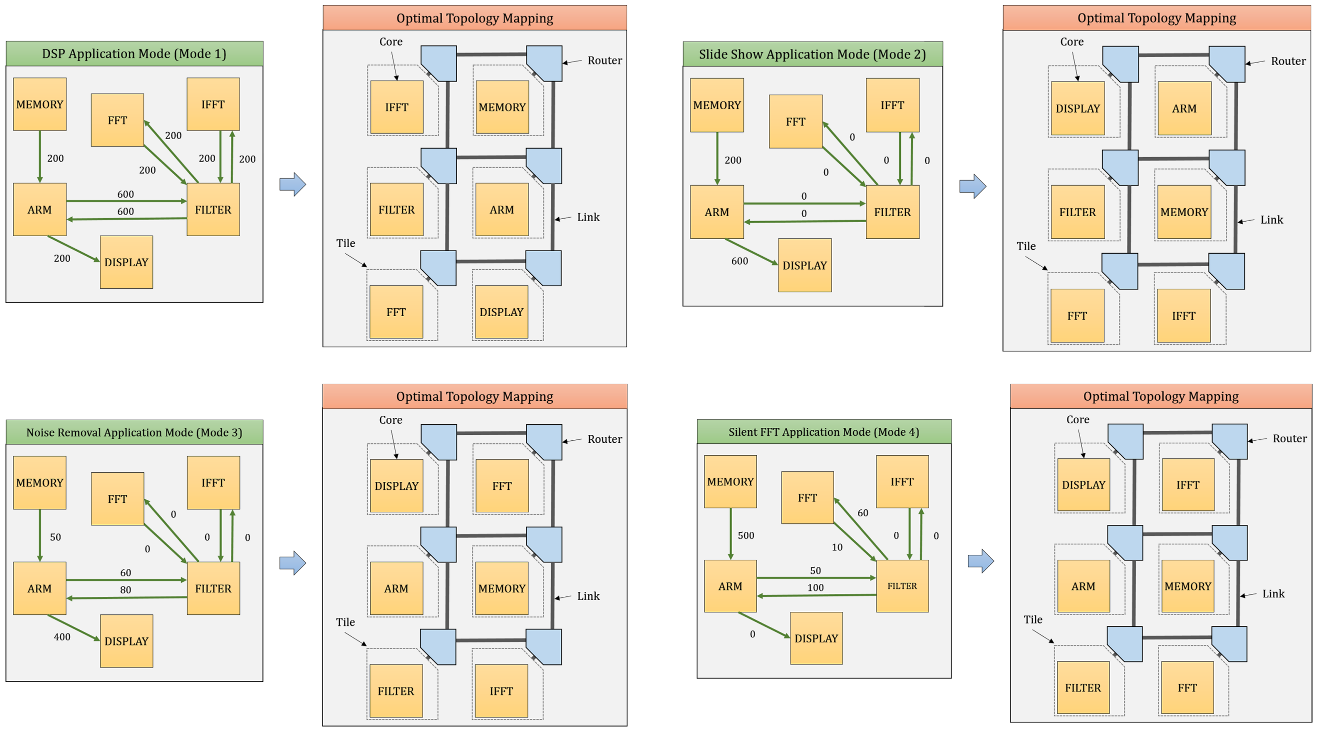

DSP Application Mode (Mode 1)
Optimal Topology Mapping
Slide Show Application Mode (Mode 2)
Noise Removal Application Mode (Mode 3)
Silent FFT Application Mode (Mode 4)
MEMORY
FFT
IFFT
ARM
FILTER
DISPLAY
Core
Router
Link
Tile

Figure 15: Figure depicts how different application modes would change the mapping of cores to tiles.

is surrounded by 4 switches in each direction and a switch in the router position itself. These switches are configured in case the packets need to bypass the router completely.

The advantage of using switches as opposed to router for bypassing is that switched consume $1/5^{th}$ power of a typical router and the packets traverse through switches much quicker than in router (within $\approxeq 0.5cycle$) [11].

Another method of network topology reconfigurability is through virtual hardware components where each VHC is loaded from the memory onto field of reconfigurable components and then appropriate connections

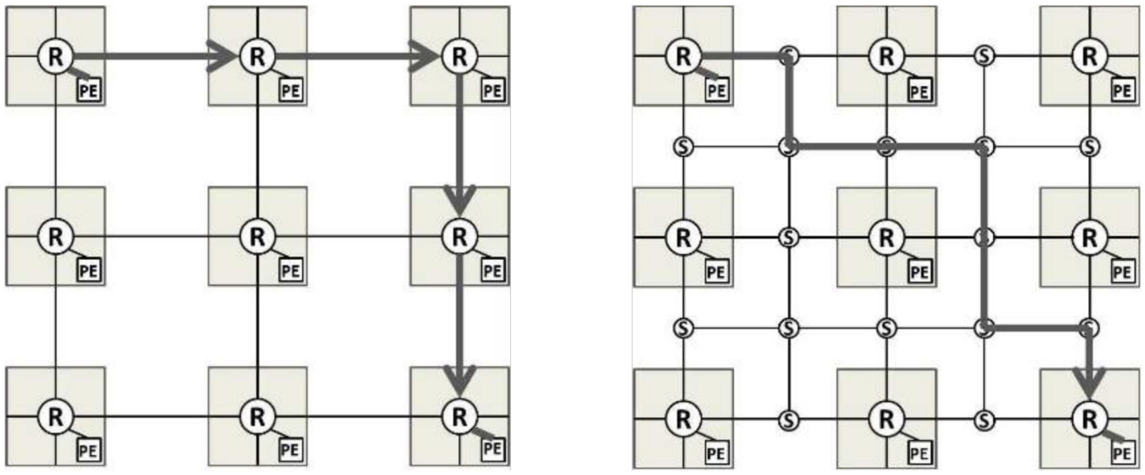


Figure 16: Installing switches on all corners of router allows reconfigurability of topology. *Left:* Traditional NoC mesh topology. *Right:* Proposed router architecture

are made.

1) **Routers surrounded by switches**

   Advantages: Good for hard-IPs and ASICs where IPs are fixed and the network needs to adapt. Allows higher frequency of operation because of ASIC.

   Disadvantages: Requires extra switch hardware, extra configuration hardware and dissipates power. Latency increases and operating frequency reduces because of unpredictable data path.

2) **VHCs with loading controller**

   Advantages: Since components are stored in the memory as virtual hardware components, component redundancy and life increases. Because components are directly connected to neighboring routers, no additional power is consumed by switches while latency is reduced.

   Disadvantages: Due to the fact that virtual hardware components are stored in memory, larger memory is requires to fit all virtual hardware components. Also in order to meet configuration time constraints, faster and expensive memory might be required.

3) **Self-loading VHCs**

   Advantages: Component redundancy and network redundancy. Even if part of network is faulty, components are going to find a way around that to communicate and connect.

   Disadvantages: Due to the fact that virtual hardware components are stored in memory, larger memory is requires to fit all virtual hardware components. Also in order to meet configuration time constraints, faster and expensive memory might be required. Additionally, signaling and strobing between different hardware components after setup is going to cause am additional set-up &

synchronization delay.

F. Buffer Depth

NoC designs rely on compromising among latency, power dissipation, or energy. The balance is normally defined at design time. However, setting all parameters at design time can lead to either excessive power dissipation or a higher latency. When large buffer sizes are used, there is a guarantee of reasonable performance during the execution of different applications but there is also router total power dissipation. Moreover, sizing buffers for the worst case latency leads to extra dissipation for the mean case (which is more frequent) [12].

Matos et al. proposed the use of a reconfigurable router in which buffer slots are dynamically allocated for optimizing power and increasing energy usage while sustaining high performance even when the application changes the communication pattern. There is no requirement for oversizing buffers to guarantee performance [12].

The proposed solution has a heterogenous router where each channel has a different buffer size. All buffers are not used all the time. A channel can lend part or all of its buffer slots according to the requirements of the neighbouring buffers [12].

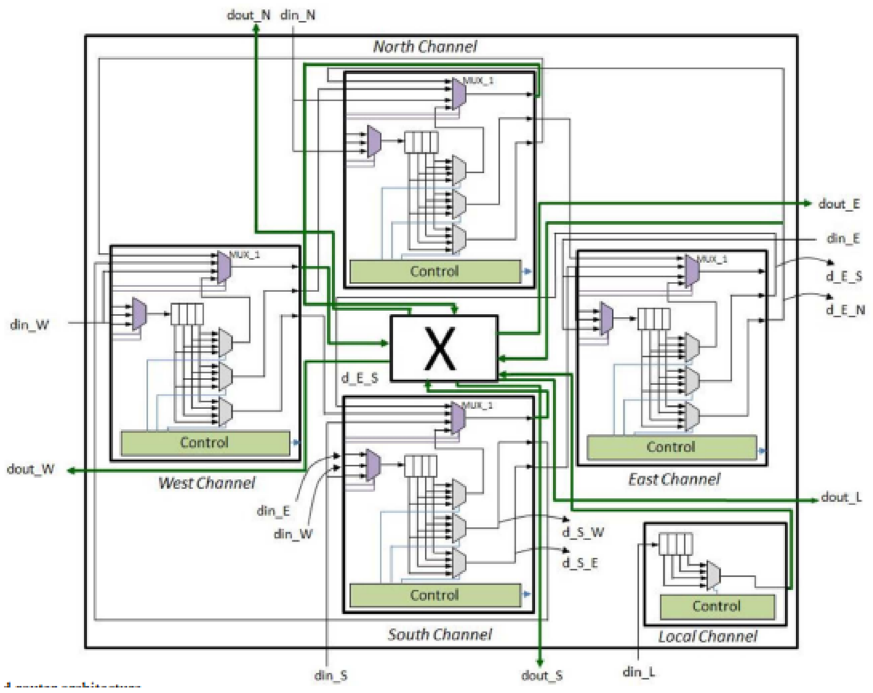


Figure 17: Architecture of the proposed router. Image Source: [12]

Experimental and simulation results done using four applications (MPEG4. MWD, VOPD and Xbox) showed that the new router was able to achieve a reduction of approximately 25% in power consumption for the worst case and 52% for the best case. When compared with the ViChaR architecture, the proposed router achieved a power reduction of about 78%. Additionally, the reconfigurable router achieved the same performance as the homogeneous router with a buffer depth that was 64% smaller [12].

Multiple physical links share memory among themselves using the multi-port memory in the link-sharing method of the wormhole routed network on chip architecture. Even though there is improvement in the communication performance, additional hardware costs are incurred [13].

Fukase et al. proposed a partial sharing method of memory by two physical links. The limited sharing by two links led to reduction of hardware costs as the shared memory has fewer ports [13].

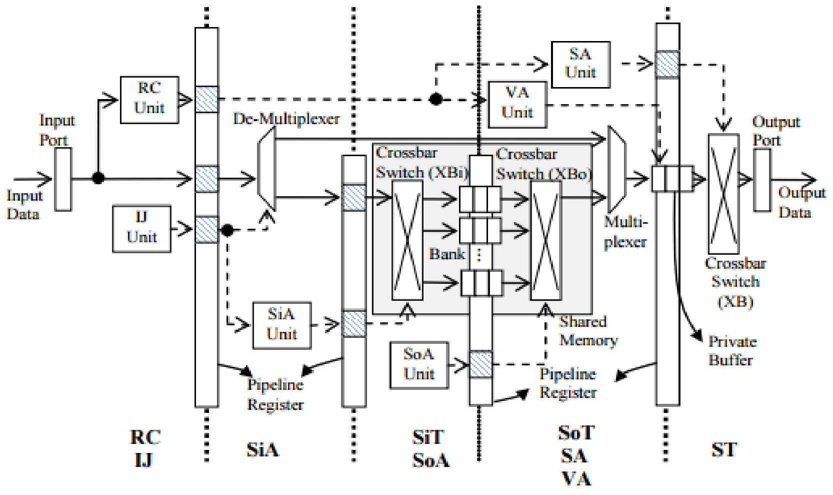


Figure 18: Block diagram of proposed method. Image Source: [13]

Performance evaluation was done by calculating the hardware cost and evaluating communication performance by software simulation. The results showed there was a reduction in hardware cost but a slight decline in communication performance. The proposed method is believed to still be suitable for various implementation schemes [13].

To address the problems of power and area in NoC architecture, Selvaraj and Kashwan proposed a low power and low area NoC architecture with reconfigurable adaptive storage buffer for data transmission and storage [14].

The proposed NoC architecture introduced repeaters between routers. The control circuit performed the function of reconfigurable adaptive routing during network traffic [14].

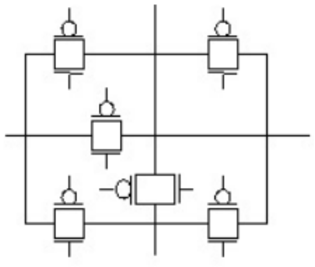


Figure 19: Adaptive Buffer Image Source: [14]

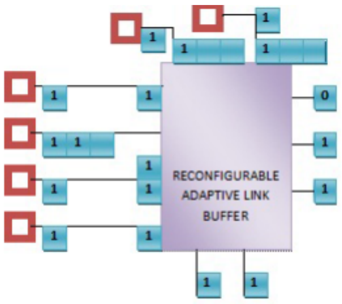


Figure 20: Reconfigurable adaptive buffer simulated using Xilinx. Image Source: [14]

The proposed solution was implemented using mesh network topology. Simulation results showed that it saves power, area and decreases the latency. Power minimization occurred using the low power gating technique. The area was minimized using the zero buffer technique [14].

Latency is reduced in an irregular mesh Network-on-Chip architecture using a heterogeneous adaptable router. Routers with large input buffers lead to excessive power dissipation and high hardware overheads.Small buffers cause high latency [15].

In the router proposed by Umamaheshwari et al. input buffers can be dynamically allocated which allows latency to be reduced. It uses small buffers and performs better as a large fixed size buffered router. Every input channel in the router can lend/borrow buffer slots to/from neighbors to get the bandwidth needed [15].

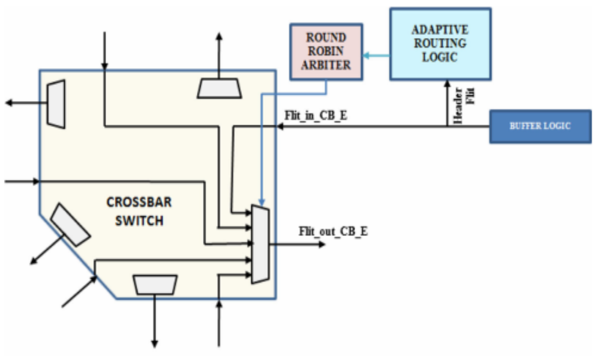


Figure 21: Router architecture. Image Source: [15]

Evaluations were performed using two 4 x 4 irregular mesh NoCs. One of them involved the proposed router and the other a static buffer router. More complex evaluations were also performed using 8 x 8 irregular mesh NoCs. Results showed that there was 30. 42% reduction in average latency and 18.33% improvement in throughput for synthetic traffic patterns. It also consumes 53% less power and achieves the same performance as a static buffer router (with a buffer depth of 10) while occupying half of its silicon area [15].

G. Router Pipeline

In a normal router, a flit goes through 4 router pipelined stages. The number of stages through which a head flit goes are different than tail or body flit. Similar router policy can be applied when a network operated in such a mode that there are known sources and destinations. Router pipeline stages such as Virtual Channel Allocation (VA) or Switch Allocation (SA) can be entirely skipped and the flit is allowed to traverse.

Pingqiang Zhou *et. al.* [16] propose a flexible-pipeline router, where the number of pipeline stages can be dynamically reconfigured. They propose to reduce the number of pipeline stages when scaling down the NoC operating frequency.

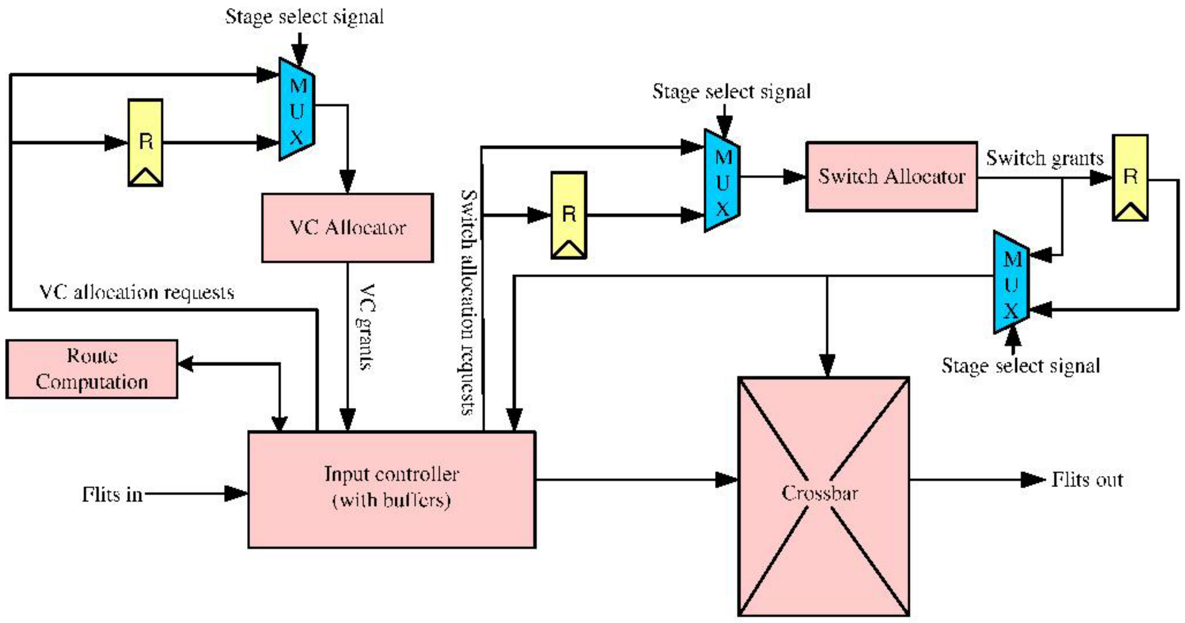


Figure 22: Reconfigurable router pipeline architecture. Image Source: [16]

IV. Conclusion

Reconfigurable computing bridges the gap between hardware and software. It guarantees better performance than software and greater flexibility than hardware. The NoC domain has benefited greatly from reconfigurable computing. Almost all the reconfigurable solutions proposed for the NoC domain gave better results for latency, throughput, bandwidth utilization, etc.

Whenever in hardware, there is a question of reconfigurability, flexibility or adaptability, it is seen that memory tables, lists and registers are used for dynamic reconfiguration. Memory is an important factor which makes a stupid machine intelligent. In routers, memory is needed to remember which flit or channel is assigned to which port. This technique allows dynamic allocation and sharing of resources to save time and space. The fact is, memories however are not cheap space-wise and cost-wise. It takes large area and consumes high power. Somewhere between too much memory and less memory there is a thin line of optimization to have a balance between chip-area, power, and reconfigurability. It all depends on the constraints and specifications.

Appendix A
Interconnection Networks

An interconnection network is a programmable system that transports data between terminals.[17]

Interconnection networks are present in almost all digital systems comprising of two or more components connected to each other. The most common place to find an interconnection network is in computers systems where processors are connected to memories and I/O devices to I/O controllers. They are the limiting factor in performance of many systems. The interconnection network between processor and memory largely determines the memory latency and and memory bandwidth, which are the two key performance factors in a computer system.[17]

A. *Terms & Concepts*

Fabric: The interconnection network medium which allows two or more terminals to connect to each other. This is sometimes collection of routers connected to each other using a particular network topology or it can be a single/collection of multi-port switches.

Capacity: The number of ports and data rate of the network. It is the rate at which information can be reliably transmitted over a interconnection network.

Latency: It is the time between a message is sent from the sender and the response arrives at the sender. In other words, it is how *late* the response is.

FLITS (**FL**ow Control Un**ITS**): Before transmission of packets, packets are split into smaller units called flits. Then these flits actually travel in the network.

B. *Flow Control*

Flow control in a network basically defines set of rules to move data from the sender to the receiver.

Characteristic	**STALL/GO**	**T-Error**	**ACK/NACK**
Buffer Area	$2N + 2$	$> 3M + 2$	$3N + k$
Logic Area	Low	High	Medium
Performance	Good	Good	Depends
Power(est.)	Low	Medium/High	High
Fault Tolerance	Unavailable	Partial	Supported

Table I: Flow control protocols at a glance. [18].

STALL-GO Flow Control: In this type of flow control, the receiver informs the sender whether it is ready to receive more data. This is a type of explicit flow control where the control decision is taken at the receiver's end.

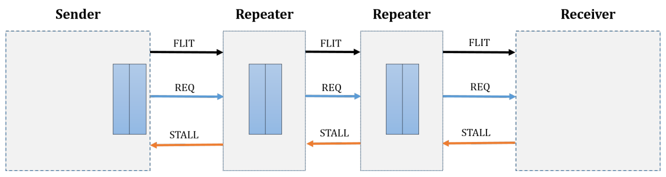


Figure 23: If not enough buffer space is available in the next router/repeater, the stall signal is send to the previous router/repeater.

The router is modified to have two more signal wires other than phit length. One signal forward indicating presence of new data and another backward informing condition of input buffers either Filled('STALL') or Empty('GO').

However, the main disadvantage of STALL/GO flow control is that no fault handling is done should any flit get corrupted. In case of a corrupted flit, the error handling task is delayed over to some higher-level protocol.

T-Error Flow Control: This type of flow control is similar to STALL-GO flow control except that an extra resynchronization stage is added between the end of the link and the receiving switch. T-Error flow control often increases the operating frequency of the link. This requires an additional resynchronization stage near the end of the link which is done with the help from combination of clock and delayed clock signals.

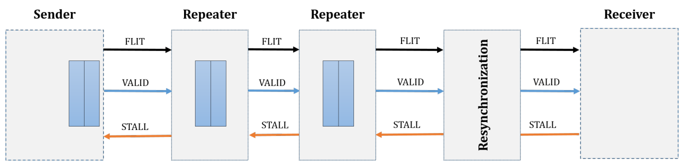


Figure 24: A Resynchronization stage is added to synchronize data signals with valid signals by use of clock and delayed clock signals.

ACK/NACK Flow Control: In this type of flow control, a copy of the flit is kept at the sender's end and the flit is sent. If the receiver acknowledges that it received the packet, the flit is discarded and the next one is sent as before procedure.

It can be implemented in two ways, end-to-end and switch-to-switch. In end-to-end, the copies of flits are kept at the sender side and the final receiver acknowledges reception of flit. On the other hand, in switch-to-switch, the copies of flits are kept in any sending switch and the acknowledgement of flit received is sent back from the next switch.

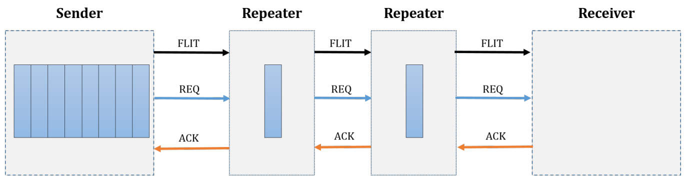


Figure 25: The repeaters on the link can be simple registers while the number of buffer requirements for the sender and receiver side is $2N + k$ buffers to guarantee maximum throughput. N: Number of repeaters.

C. Routing

Routing mechanisms determine how packets/flits move from source/sender to sink/destination. There are basically two categories of routing algorithms.

A good routing algorithm must prevent any potential deadlock, starvation, and and livelock situation of packets/flits in the network.

Deadlock situation may arise in NoC when any circular waiting path is generated from the routing algorithm. In order to prevent any deadlock, at least one of the cyclic dependency must be broken. This is done by either creating virtual channels or by restricting the packet movement.[19]

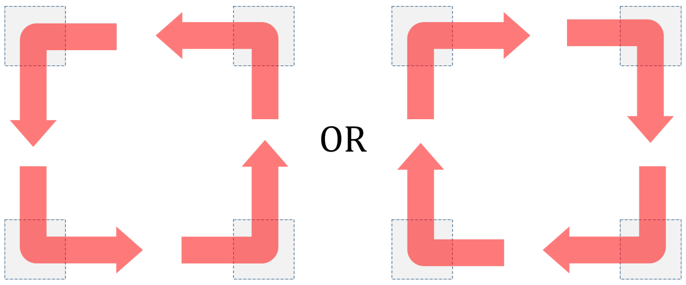


Figure 26: Each of the packet in these routers is trying to make a turn but the resources are already occupied by another packet. This causes an indeterminate waiting period and none of the packets are able to move. This is called as deadlock.

***Deterministic Routing**:* The route taken by the packet in reaching from source/sender to sink/destination is pre-determined and known. This route is fixed an each source and sink pair have a unique route which is constant throughout the run time.

Examples of deterministic routing include:

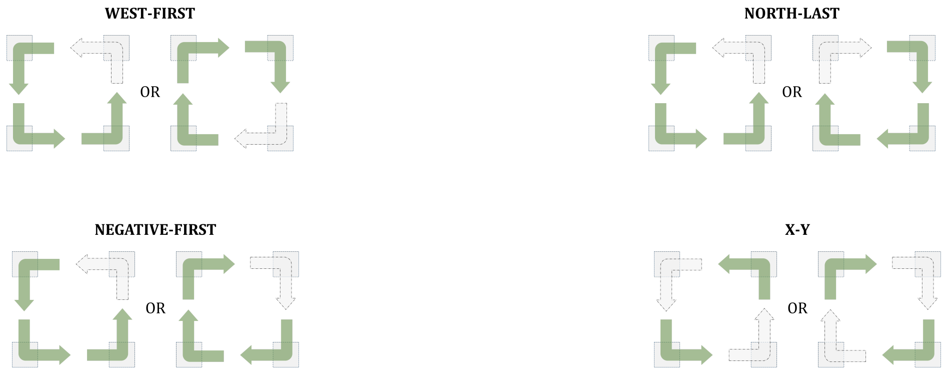


Figure 27: Movement restrictions such as West-First, North-Last, Negative-First, and X-Y prevent deadlocks from happening. This is because one or more of the movements in cycle cannot happen thus, the dependency cycle is broken.

1. ***Dimension ordered routing (X-Y) or (Y-X):*** The packet first moves in one dimension and reaches the destination's row or column and then continues to move in other dimension to reach destination. In this type of network, each router knows where other routers are and can determine if the packet's destination lies at west, east, south or north of it. An example of X-Y dimension ordered routing is shown in Figure 28.

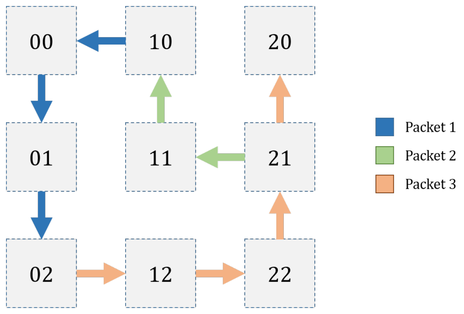


Figure 28: The figure shows how a packet is routed in 3x3 mesh based network with X-Y routing algorithm. Each packet travels in X dimension to reach destination's column and then in Y dimension to reach destination.

2. ***Destination-Tag Routing:*** Each packet from the sender/source to the receiver/sink is tagged with a destination address in the header. Based on this tag, the intermediate nodes/routers determine the path packet has to take using the routing table. An example of this type of routing is a 4-ary 2-fly butterfly network shown in Figure 29

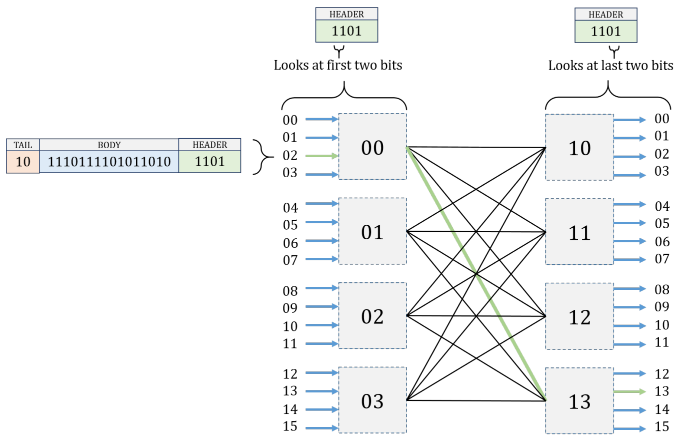


Figure 29: The figure shows how a packet is routed from address 02 to address 13. The first array of router look at the first two bits of address in packet and the second array of routers look at last two bits of address in packet.

Oblivious Routing: Oblivious literally means being "unaware of". In this type of routing, a packet is routed without regard for the current state of the network. If a packet is to be send from A to B, another node is chosen at random by the sender A and the packet is sent first to this randomly chosen node. Then this node forwards the packet to node B. The figure 30 shows the process of Oblivious Routing.

This type of routing causes the load on all channels to distribute and spread out toward the network rather than sticking to one side of the network. The advantage of oblivious routing is that the channel loads and traffic patterns are linearly related. Hence it makes it easy to compute ideal throughput and traffic pattern from given channel load.[17]

However, because of the fact that the source router can select any intermediate router to send the message first to, the traffic is widespread over the whole network. To preserve locality of the network, the intermediate router is chosen from a defined region around the sender with minimal path. This is called as Minimal Oblivious Routing.

Adaptive Routing: Adaptive routing algorithms use information about the network in real-time to adapt and change how different packets are routed on the network.

1. ***Load Balanced Adaptive Routing:*** The algorithm tries to equalize the traffic load on all the links equally. An example of load balanced adaptive routing is given in [20] where the authors try to balance the load by the use of Ant Colony Optimization (ACO). ACO is a biologically inspired from ant colonies and how they locate optimal paths by use of pheromones, attractants, and accumulation of paths. They propose a NoC version of ACO which was used extensively in Wide Area Networks

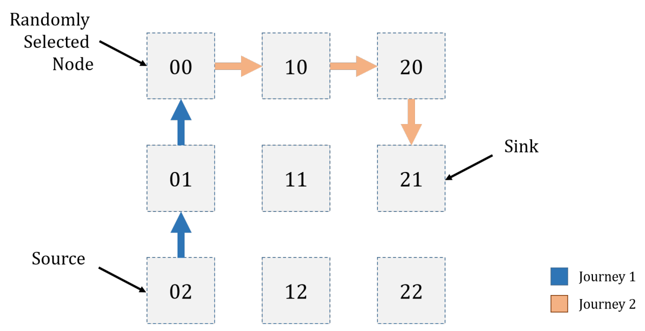


Figure 30: The figure shows how a packet is routed from address 02 to address 21 using Oblivious Routing Algorithm.

and call it Regional ACO-based Cascaded Adaptive Routing (RACO-CAR). This technique eliminates tables which contain redundant information, shares routing tables with neighboring routers and merge information, and cascades routing to distribute the load in the network.

2. ***Fully Adaptive Routing:*** This type of routing algorithm adapts to the congestion and blocking in the network and re-routes the packets away from congested or blocked areas. An example of fully adaptive routing algorithm is given in [21]. The authors propose *FreeRider*: A non-local adaptive congestion aware algorithm where the congestion information of whole network (non-local) is used to make routing decisions. Rather than using the Congestion Propagation Network (CPN) to propagate congestion information, but instead they propose to use free bits in header flit to carry congestion information. This improves the throughput, shortens the latency, and reduces the power consumption.
3. ***Minimal Adaptive Routing:*** Minimal route searching algorithms look for the shortest/minimal path to the destination before sending each packet.

The major challenge in adaptive routing is the fact that the route for each packet is decided in real-time just before a packet is sent from one node to another which can cause livelocks. Livelocks are different from deadlocks because in livelock, the packets are still moving but in a loop. On the other hand, in deadlock, all movement of packets is blocked in the loop.

Appendix B
Link Frequency Reconfiguration in NoCs

Nowadays NoCs are designed by considering the worst case scenarios of the CMOS technology. But the real-time parameters are quite different from the worst-case assumed at the design time. If the network is designed in such a way that it can find out optimal operating conditions using the current real-time parameters (eg. error rate, gate-delay etc.), then it will increase the energy efficiency of the network. This gives rise to self-calibrating networks which are intelligent enough to modify their operating parameters (such as operating voltage, contention, error protection technique etc.) to reduce energy consumption and maximize performance.

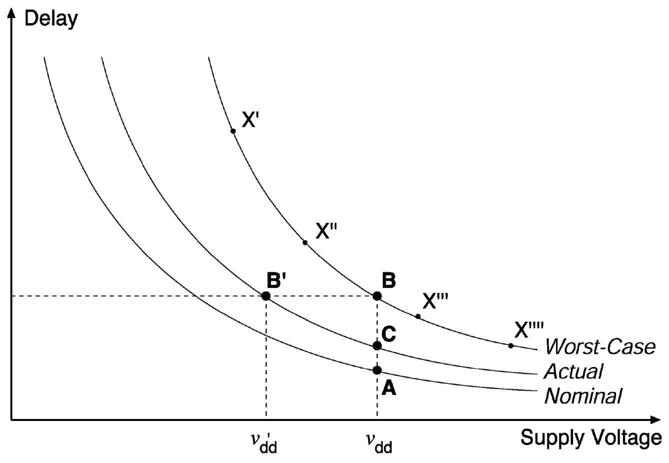


Figure 31: Assuming the worst-case scenario at design time forces a much higher V_{dd} than actual V_{dd} required. Furthermore, if the traffic is low, higher delay in packet delivery can be afforded at the advantage of low energy requirement. [22]

Self-calibrating NoCs aim to maintain an optimal trade-off between energy, performance, and reliability. Rather than relying on traditional CMOS conservative worst-case assumptions, these NoCs adjust their operating parameters to run-time actual conditions in order to achieve maximum performance and optimal reliability through minimum energy consumption.

However, self-calibration requires intelligent hardware error detection and correction

A. Self-calibrating link

G. D. Micheli et al proposed to separate the frequency of link from the frequency of router.[22] An extra FIFO buffer is required at the output of router. This decouples the link from router and allows the link to operate at a different frequency than router.

The controller takes error rate signal from the decoder at receiver's side and the traffic signal from the output FIFO buffer. The controller decides on the frequency required in order to meet the application requirements and then according to frequency required, input traffic, error rate, and reliability constraint the appropriate channel voltage is determined. [23]

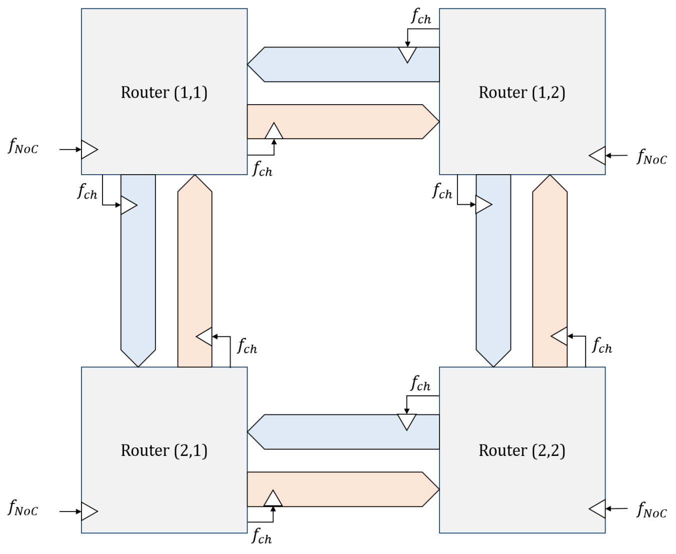


Figure 32: The figure shows self-calibrating Network-on-Chip built with self-calibrating links. The links operate at a different frequency than the network.

It is possible that the decoder can declare a word correct while it is corrupted. The probability of such a thing happening is called as *residual word error rate*. The controller has to ensure that this probability is below maximum tolerable limit.

The optimization problem for the controller can be split into two steps:

1. Choose the lowest frequency that meets the bandwidth requirements.
2. For the chosen frequency, pick the lowest voltage which does not cause detected word errors.

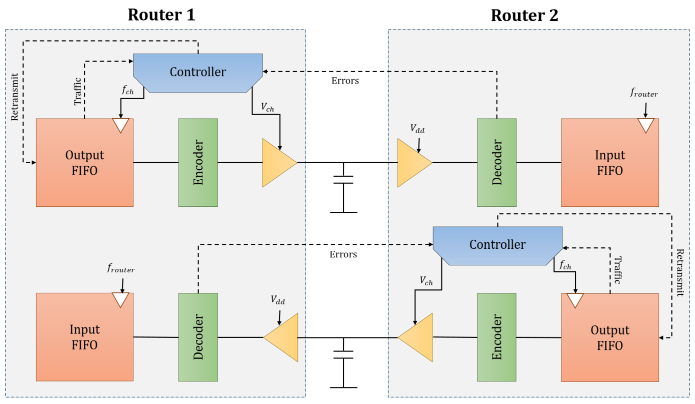


Figure 33: The figure shows bi-directional self-calibrating link between two routers. If an error is detected at the receiving end, the flit is re-transmitted. The [22]

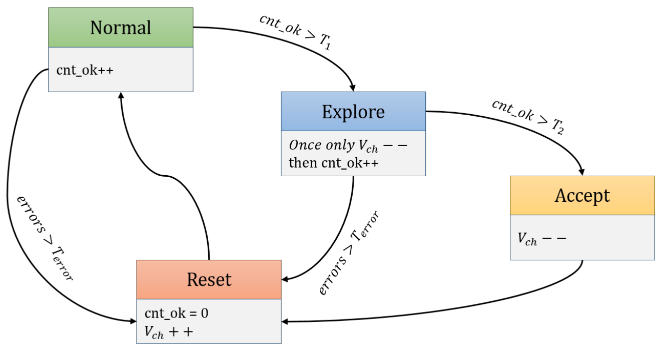


Figure 34: State diagram for controlling the voltage of the link via incremental/decremental digital control.

In *Normal* state, if the correct transmissions occur, the ok_counter is incremented and if the ok_counter goes above the Threshold 1 (T_1), then the program enters the *Explore* mode. Here the voltage is decremented once and then further errors are counted. If the error goes above Threshold 2 (T_2), then the current voltage is accepted as the correct one and the machine resets. In *Normal* or *Explore* states, if errors occur, the voltage is increased and *Normal* state is re-entered.[22]

Including a controller at each output port requires additional hardware leading to additional die area for the router. If the application requires minimum power consumption and is able to sacrifice some die area for it then self-calibrating links are a viable option.

B. Contention-Aware Energy Management

When packets of different flows compete for the same link, this is called as *contention*. *Jian-Jun Han et al* in their paper[24] propose a more intelligent algorithm for Dynamic Voltage and Frequency Scaling (DVFS) which takes into consideration the amount of contention occurring in the network. They used deterministic XY routing on a mesh based homogenous VFI NoC with partial VCT (Virtual Cut Through) flow control. The formula for total power consumption then becomes:

$$Pow_{consumed} = \sum_{i=1}^{M} s_i \times Pow_{dyn.}(f_i(P_i)) + Pow_{leak.}(f_i(P_i)) \tag{1}$$

Where $Pow_{consumed}$ is power consumed, $Pow_{leakage}$ is leakage power, s_i is the state of processor. If processor is active, $s_i = 1$ else $s_i = 0$. M is total number of processors on network. $f(P_i)$ is the frequency of the processor i. The complete derivation of equation is explained in [**?**].

CA-TIMES-Quick algorithm was introduced where a task is assigned to the core first and then all communication routes are determined. In this algorithm, the tasks are organized according to their priority.

The algorithm can be divided into two main components.

1) Task Mapping: A suitable core is chosen where there is minimum distance between the current task and it's predecessor. If there are multiple options for locations of core, then the core is assigned to the same VFI where it's maximum predecessor reside. If the core is still not decided, then the core with lowest index is chosen.

2) Prioritization of Edges: For any two data transfers(edges) ($edge_{m \to a}$ and $edge_{n \to a}$), if finish time of task m on processor x is lower than finish time of task n on processor y, then $edge_{m \to a}$ has priority over $edge_{n \to a}$. If the task finish times are equal, then the task priority decides the edge priority.

3) Route Search for an Edge: Deadlock-free Breadth-First-Search algorithm is used to find paths for each edge from core to core. The path with the earliest finish time of edge through it's last link is chosen.

REFERENCES

[1] G. E. Moore, “Cramming more components onto integrated circuits,” *Electronics*, vol. 38, no. 8, April 1965.

[2] S. Tade, H. Matsutani, H. Amano, and M. Koibuchi, “A metamorphotic network-on-chip for various types of parallel applications,” in *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, July 2015, pp. 98–105.

[3] J. Delorme, J. Martin, A. Nafkha, C. Moy, F. Clermidy, P. Leray, and J. Palicot, “A fpga partial reconfiguration design approach for cognitive radio based on noc architecture,” in *Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on*, June 2008, pp. 355–358.

[4] T. Bjerregaard and S. Mahadevan, “A survey of research and practices of network-on-chip,” *ACM Comput. Surv.*, vol. 38, no. 1, Jun. 2006. [Online]. Available: http://doi.acm.org/10.1145/1132952.1132953

[5] D. Wang, M. Koibuchi, T. Yoneda, H. Matsutani, and H. Amano, “A dynamic link-width optimization for network-on-chip,” in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2011 IEEE 17th International Conference on*, vol. 2, Aug 2011, pp. 106–108.

[6] K. Latif, A.-M. Rahmani, E. Nigussie, T. Seceleanu, M. Radetzki, and H. Tenhunen, “Partial virtual channel sharing: A generic methodology to enhance resource management and fault tolerance in networks-on-chip,” *Journal of Electronic Testing*, vol. 29, no. 3, pp. 431–452, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10836-013-5389-5

[7] M. Oveis-Gharan and G. Khan, “Efficient dynamic virtual channel organization and architecture for noc systems,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[8] M. O. Gharan and G. N. Khan, “Packet-based adaptive virtual channel configuration for noc systems,” *Procedia Computer Science*, vol. 34, pp. 552 – 558, 2014, the 9th International Conference on Future Networks and Communications (FNC’14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC’14)/Affiliated Workshops. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914009247

[9] N. Kavaldjiev, G. Smit, and P. Jansen, “A virtual channel router for on-chip networks,” in *SOC Conference, 2004. Proceedings. IEEE International*, Sept 2004, pp. 289–293.

[10] G. Jiang, Z. Li, F. Wang, and S. Wei, “Mapping of embedded applications on hybrid networks-on-chip with multiple switching mechanisms,” *Embedded Systems Letters, IEEE*, vol. 7, no. 2, pp. 59–62, June 2015.

[11] P. Darbani and H. Zarandi, “A reconfigurable network-on-chip architecture to improve overall performance and throughput,” in *Electrical Engineering (ICEE), 2014 22nd Iranian Conference on*, May 2014, pp. 943–948.

[12] D. Matos, C. Concatto, M. Kreutz, F. Kastensmidt, L. Carro, and A. Susin, “Reconfigurable routers for low power and high performance,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 2045–2057, Nov 2011.

[13] N. Fukase, Y. Miura, S. Watanabe, and M. Rahman, “The proposal of partial sharing for link-sharing method of buffer in noc router,” in *Computing and Networking (CANDAR), 2014 Second International Symposium on*, Dec 2014, pp. 567–571.

[14] G. Selvaraj and K. R. Kashwan, “Reconfigurable adaptive routing buffer design for scalable power efficient network on chip,” *Indian Journal of Science and Technology*, vol. 8, no. 12, pp. 1–9, 06 2015. [Online]. Available: http://ezproxy.lib.ryerson.ca/login?url=http://search.proquest.com/docview/1701619816?accountid=13631

[15] U. S, M. D, and R. PERINBAM J, “Runtime buffer management to improve the performance in irregular network-on-chip architecture,” *Sadhana*, vol. 40, no. 4, pp. 1117–1137, 2015. [Online]. Available: http://dx.doi.org/10.1007/s12046-015-0378-2

[16] P. Zhou, J. Yin, A. Zhai, and S. Sapatnekar, “Noc frequency scaling with flexible-pipeline routers,” in *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, Aug 2011, pp. 403–408.

[17] W. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*, 1st ed. Morgan Kaufmann, December 2003.

[18] M. H. C. P. R. O. K. S. D. Mieszko Lis, Keun Sup Shim, “DARSIM: a parallel cycle-level noc simulator.” [Online]. Available: http://people.csail.mit.edu/devadas/pubs/pdarsim.pdf

[19] P. Ghosal and T. Das, “Fl2star: A novel topology for on-chip routing in noc with fault tolerance and deadlock prevention,” in *Electronics, Computing and Communication Technologies (CONECCT), 2013 IEEE International Conference on*, Jan 2013, pp. 1–6.

[20] E.-J. Chang, H.-K. Hsin, C.-H. Chao, S.-Y. Lin, and A.-Y. Wu, “Regional aco-based cascaded adaptive routing for traffic balancing in mesh-based network-on-chip systems,” *Computers, IEEE Transactions on*, vol. 64, no. 3, pp. 868–875, March 2015.

[21] S. Liu, T. Chen, L. Li, X. Li, M. Zhang, C. Wang, H. Meng, X. Zhou, and Y. Chen, “Freerider: Non-local adaptive network-on-chip routing with packet-carried propagation of congestion information,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 8, pp. 2272–2285, Aug 2015.

[22] F. Worm, P. Ienne, P. Thiran, and G. D. Micheli, “A robust self-calibrating transmission scheme for on-chip networks,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 126 – 139, January 2005.

[23] G. D. Micheli and L. Benini, *Netwroks on Chips*. Elsevier Inc., 2006.

[24] J.-J. Han, M. Lin, D. Zhu, and L. Yang, “Contention-aware energy management scheme for noc-based multicore real-time systems,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 3, pp. 691–701, March 2015.