Cairo University

DESIGN OF BUFFER-LESS THREE DIMENSIONAL (3D) NETWORK ON CHIP (NoC) ROUTERS

By

Marwa Mohamed Zaky Shaheen

A thesis submitted to the
Faculty of Engineering at Cairo University
in Partial fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Faculty of Engineering, Cairo University
GIZA, EGYPT
2020

DESIGN OF BUFFER-LESS THREE DIMENSIONAL (3D) NETWORK ON CHIP (NoC) ROUTERS

By

Marwa Mohamed Zaky Shaheen

A thesis submitted to the
Faculty of Engineering at Cairo University
in Partial fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Prof. Dr. Mohamed Riad El-Gonemy	**Prof. Dr. Hossam A. H. Fahmy**
Professor Electronics and Communications Engineering Department Faculty of Engineering	Professor Electronics and Communications Engineering Department Faculty of Engineering

Associ. Prof. Dr. Hassan M. H. Mostafa

Assistant Professor
Electronics and Communications
Engineering Department
Faculty of Engineering

Faculty of Engineering, Cairo University
GIZA, EGYPT
2020

DESIGN OF BUFFER-LESS THREE DIMENSIONAL (3D) NETWORK ON CHIP (NoC) ROUTERS

By

Marwa Mohamed Zaky Shaheen

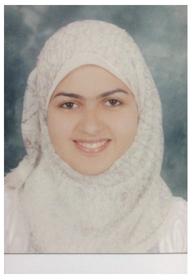
A thesis submitted to the
Faculty of Engineering at Cairo University
in Partial fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the
Examining Committee

Prof. Dr. Mohamed Riad El-Gonemy	**Thesis Main Advisor**
Prof. Dr. Hossam A. H. Fahmy	**Advisor**
Associate Prof. Dr. Hassan M. H. Mostafa	**Advisor**
Prof. Dr. Ahmed Hussein M. Khalil	**Internal Examiner**
Prof. Dr. ElSayed Mostafa Saad - Helwan University	**External Examiner**

Faculty of Engineering, Cairo University
GIZA, EGYPT
2020

Engineer's Name:	Marwa Mohamed Zaky Shaheen	
Date of Birth:	20/7/1992	
Nationality:	Egyptian	
Email:	marwa.92.shaheen@gmail.com	
Phone:	01118041720	
Address:	Electronics and Communication Engineering Cairo University	
Registration Date:	20/9/2014	
Awarding Date:	-/-/2020	
Degree:	Master of Science	
Department:	Electronics and Communication Engineering	
Supervisors:	Prof. Dr. Mohamed Riad Al-Gonemy Prof. Dr. Hossam A. H. Fahmy Dr. Hassan M. H. Mostafa	
Examiners:	Prof. Dr. Mohamed Riad Al-Gonemy	(Thesis main advisor)
	Prof. Dr. Hossam A. H. Fahmy	(advisor)
	Dr. Hassan M. H. Mostafa	(advisor)
	Prof. Dr. Ahmed Hussein M. Khalil	(Internal examiner)
	Prof. Dr. ElSayed Mostafa Saad Helwan University	(External examiner)



Title of Thesis:

Design of Buffer-less Three Dimensional (3D) Network on Chip (NoC) Routers.

Keywords:
network on chip (NoC)

Summary:
In this work brief introduction to network on chip is included. Followed by an introduction to the architecture of CONNECT and related work in buffer-less router designs. Next, this work presents Modified CONNECT buffer-less router architecture that targets FPGA. A discussion on 3-dimensional network pros and cons is introduced followed by simulations of 3D-Modified CONNECT. Later, a brief study is introduced that discusses the optimum size and feature of 3D NoC. Finally, A CAD tool is presented that summarize the work and enables the user to configure different buffer-less 3D NoCs.

Disclaimer

I hereby declare that this thesis is my own original work and that no part of it has been submitted for a degree qualification at any other university or institute. I further declare that I have appropriately acknowledged all sources used and have cited them in the references section.

Name .... Date ....

Signature ....

Acknowledgement

I would like to express my appreciation to Dr. Mohamed Riad for his heed and patience.

My endless gratitude to Dr. Hossam Fahmy for his enormous support and indulgence. I am lucky to have such a respectful, supportive and open-minded professor. I am forever thankful for his guidance and consideration. My sincere thanks for giving me the opportunity to experience this path. Thanks for giving me this wonderful opportunity to learn.

I would like to express my thanks to Dr. Hassan Mostafa for being my mentor and for giving me the opportunity to learn. I would like to express my thanks for his patience through these five years and his encouragement whenever I face any difficulty. I would like to thank him for his sincere advices and insightful comments. Finally, my thanks and love to mom, dad and my awesome brothers.

Table of Contents

List of Tables

List of Figures

Nomenclature

ASIC	Application Specific Integrated Circuit
BRAM	Block RAM
CLB	Configurable Logic Blocks
DRAM	Distributed RAM
FF	Flip Flop
FOM	Figure of Merit
FPGA	Field Programmable Gate Arrays
NoC	Network on Chip
NRE	Non-recurring Engineering Cost
PDR	Partial Dynamic reconfiguration
SoC	System on Chip
TCP	Transmission Control Protocol
VC	Virtual Channel

Abstract

As Chip Multiprocessors (CMPs) scale to tens or hundreds of nodes, the interconnect becomes a significant factor in cost, energy consumption and performance. Energy efficiency of the underlying communication framework plays a major role in the performance of multi-core systems. Recent work proposes buffer-less deflection routing as a cost effective alternative. First, this work presents literature review on the conventional on chip router architecture followed by a literature review on buffer-less router designs. Then, buffer-less routing algorithms are discussed. The main contribution of this thesis work is the development of a buffer-less router. The router is a modified version of CONNECT (CONfigurable Network Creation Tool). Modified CONNECT is efficient and lightweight. It achieves better throughput than the available buffer-less routers. It occupies less area than the available buffer-less routers. Modified CONNECT occupies 30% area less than original CONNECT and achieves competitive performance to CONNECT.

Buffer-less networks in general obtain good performance at low traffic loads as the traffic load increases, the performance of the network degrades. The performance becomes even worse as the network diameter increases. 3D architecture enhances the performance; therefore, this work is extended to three dimensions. 3D Modified CONNECT uses the same routing algorithm as CONNECT and offers better performance and less power consumption than the available 3D buffer-less routers.

A brief study is included in this work in which network parameters are evaluated in order to find the optimum match for 3D buffer-less network. The mesh size, number of tiers, number of 3D routers per each tier and the location of the 3D routers have been simulated against a figure of merit (FOM). The four mentioned parameters have been simulated for five different buffer-less 3D routers. The chosen figure of merit (FOM) equals to throughput per dynamic power, occupied area and average network latency. It is observed that the optimum values of the mentioned parameters actually depend on the architecture of each router, they could not be general to all buffer-less routers.
Finally, a new flexible CAD tool (Computer Aided Design) is proposed to evaluate different 3D buffer-less networks. It is an easy way to perform more experiments on buffer-less 3D networks. It allows the user to heavily configure the buffer-less network. The user configures the mesh size, number of tiers, the number of 3D routers and their location. Also, the user selects the buffer-less router. The traffic injection rate can also be configured. The tool allows the network to be implemented. The generated code is sent to the user as an attachment to the provided user email.

Chapter 1: Introduction

As VLSI technology scales and processing power continues to improve, inter-processor communication becomes a performance bottleneck. Interconnects also play a significant factor in cost and energy consumption. This chapter explains the difference between ASIC and FPGA, and the merits of using networks on chip on FPGA. Here is a brief discussion on basic regular network topologies, how to evaluate the performance of a network on chip, the basic concepts of switching, routing and flow control used in any network and the buffering resources available on FPGA. General speaking, FPGA has a simple design cycle what makes it more suitable for research. Also, it has shorter time to market than ASIC. ASIC has complex floor planning, because ASIC generates specific layout for a certain application. FPGA has a fixed layout, the design tends to make the best use of the floor area and the best utilization of the FPGA resources.

1.1 FPGA vs ASIC

FPGA is replacing ASIC in many applications. Though, ASIC is more efficient in power, area, performance and even the cost, FPGA is pushing ASICs out of the market. FPGA provides programmable logic during run time. It offers quick time to market. FPGA actually replaces ASIC in research. The standard cells in ASIC are faster than look up tables in FPGA. Therefore, ASIC has higher operating frequency. ASIC implements the logic efficiently occupying less resources. The standard cells of the basic gates(and, or, not) are used to build any logic. Once the logic is implemented and fabricated, it can not be changed. ASIC is suitable for high volume designs; because ASIC occupies less area resources. FPGA maps the logic truth table in the look up tables. FPGA could be reconfigured, as the look up tables are reprogrammed to hold a truth table of another logic. the system implemented on FPGA will be slower than hard implementation (ASIC design); because look up tables are slower than the standard cells. Many FPGA designs tend to pipeline their architecture in order to enhance the operating frequency. The designs in ASIC are more complex, it takes large compile time. Designs on FPGA will consume also more power and occupies larger area.

1.1.1 Cost

ASIC has lower manufacturing costs. ASIC is more efficient in the production of large number of chips. The non-recurring costs of ASIC are larger than FPGA, the total cost of ASIC decreases with mass production. FPGA has lower recurring costs; therefore, it is used in research. As shown in 1.1 the increase in the cost of ASIC forces the manufacturers either to use older technologies or to use FPGAs instead of ASIC.

1.1.2 Time to market

FPGA has lower recurring cost and shorter design cycle. The product is ready to market before ASIC products. Long time to market is a cost. The profit is much larger if you come to the market

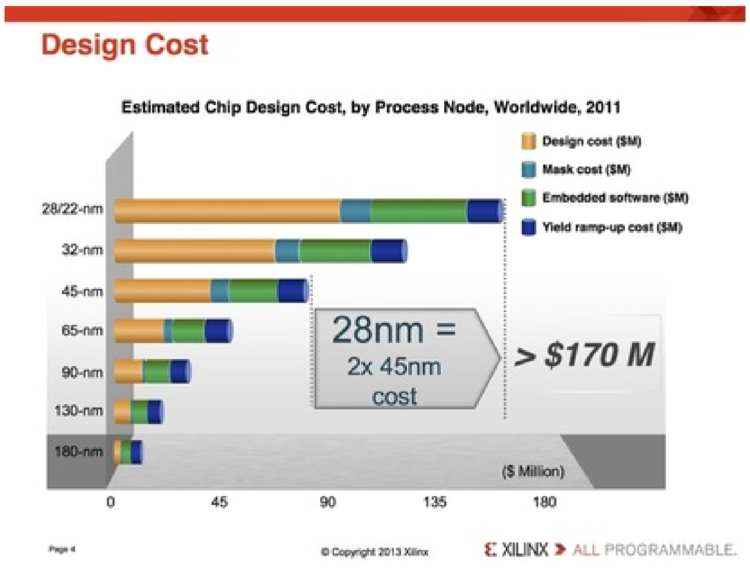


Fig. 1.1: ASIC Cost Analysis [1]

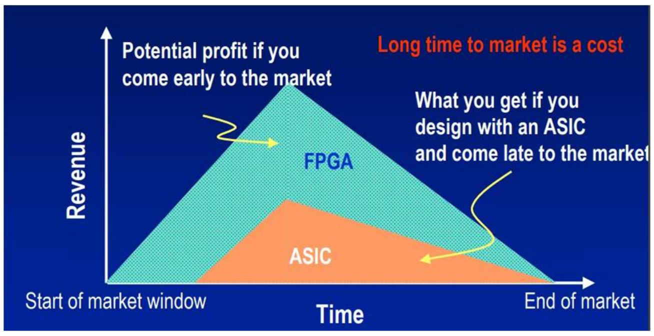

Potential profit if you come early to the market
Long time to market is a cost
What you get if you design with an ASIC and come late to the market
FPGA
ASIC
Revenue
Start of market window
Time
End of market

Fig. 1.2: Time to Market Cost [11]

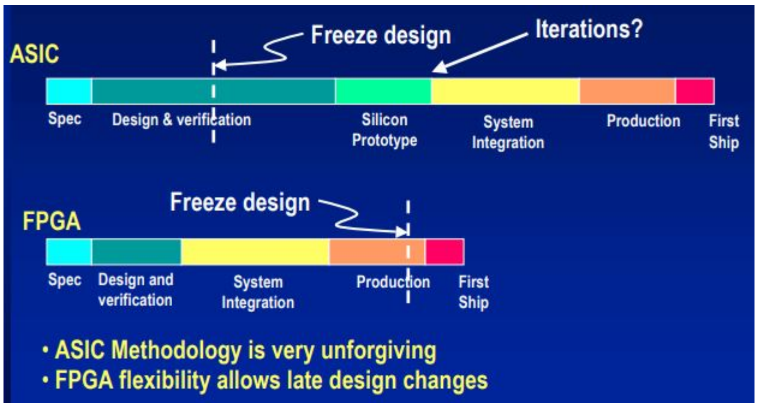
early as in Fig.1.2. Missing the market window will wipe out all the savings from development and production.

1.1.3 Reconfigurability

FPGA offers a flexible life cycle management. In ASIC once the layouts are constructed, they can not be changed. However, the design is more flexible on FPGA. FPGA offers dynamic partial configuration where look up tables could be configured while others are running. Partial reconfiguration saves the area as it programs only the needed physical resources in each phase.[24] Partial reconfiguration is achieved by loading the partial bitstream of a new design into the FPGA configuration memory. The reconfigurable portion cannot work at that time due to the incompleteness of the configuration data. It has to wait till the data is loaded. Fast applications, which need switching between multiple IPs frequently, require small reconfiguration time. The network size and the number of parallel DPRs depend on the desired reconfiguration time. Supporting more simultaneous DPRs needs more area resources such as decoupling buffers and controlling units; however, the network size does not change and the reconfiguration time does not change. The NoC based FPGA has reconfiguration time 1.25 times better than conventional SRAM based FPGAs [35].

1.1.4 Design Cycle

The design cycle is easier on FPGA. Placement and routing stages are automated. Therefore late design changes are easy and the software is extremely fast. The design is flexible. In ASIC the floor planning is complex. The process issues affect the design cycle.

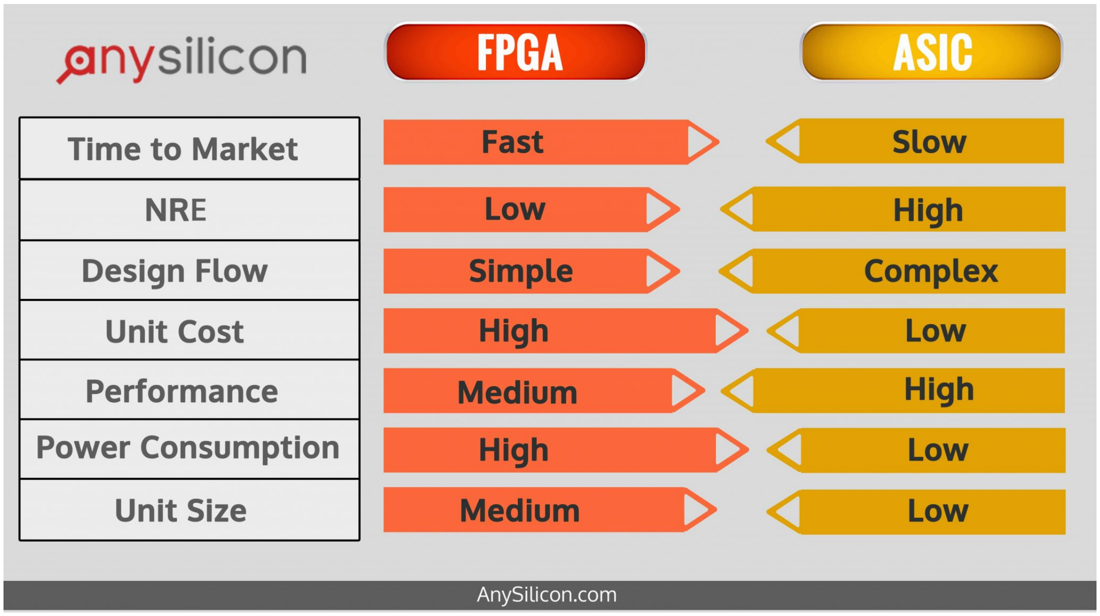


- ASIC Methodology is very unforgiving
- FPGA flexibility allows late design changes

Fig. 1.3: Design Cycle [38]

1.1.5 Summary

Here is the summary of the previous items.



anysilicon	FPGA	ASIC
Time to Market	Fast	Slow
NRE	Low	High
Design Flow	Simple	Complex
Unit Cost	High	Low
Performance	Medium	High
Power Consumption	High	Low
Unit Size	Medium	Low

AnySilicon.com

Fig. 1.4: FPGA vs. ASIC [10]

1.2 Why network on chip on FPGA

Network on chip implemented on FPGA offers component re-use and partial configuration. This is a very promising approach. The application could be changed during run time. A block could be replaced while other blocks are functioning this is called dynamic partial configuration(DPR). It opens a new way for dynamic multitasking applications. Programmed FPGA is used to provide flexibility and fast prototyping implementation. Network on chip on FPGA solves the

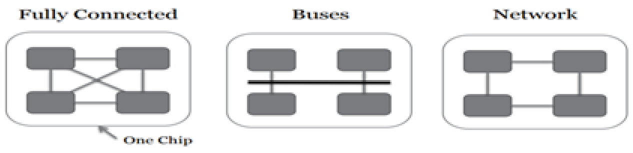


Fig. 1.5: Buses and networks

problem of frequency requirements for different peripherals connecting different IPs. However, the nature of networks perfectly suits the FPGA. The scalability and path diversity offered by the network allow parallel communication between different modules. They offer concurrent connections between nodes, parallel compilation, dynamic configuration and dynamic multitasking. The main advantage of NoC based FPGAs over the conventional SRAM-based FPGAs is the ability to perform multiple simultaneous DPRs (dynamic partial reconfigurations). It reduces the run time for high level abstraction. The re-configuration time of DPR with NoC is better than the reconfiguration time on conventional FPGAs considering applications that concern configuration time over area overhead [35].

1.3 Why network on chip

As the number of processors per chip increases, the dynamic energy consumed in the routes is higher than that inside the logic. Designing the routes between different IPs is very critical. Network scales better with the increase in the number of nodes. NoCs are suitable to connect this large number of components, due to their modular design and their scalability. The short wires used in NoC between the cores decrease the capacitance and resistance of the routes. Therefore, the delay decreases. NoC replaces the global long wires with short segmented wires and router nodes. Network on chip becomes a parallel, concurrent, scalable and modular alternative to traditional buses.
Parallel and Concurrent: In shared bus, there is only a shared link common to all nodes. It does not support any parallelism, unlike NoC which offers concurrent connections between cores and parallel compilation. A topology dictates the number of nodes and the number of alternate paths between nodes, and thus how well the network can handle contention and different traffic patterns.
Scalability: Shared buses are area efficient but buses do not scale with the increase in the capacity neither in terms of performance nor power. Network on chip is easily scalable in performance, complexity is proportional to number of nodes unlike fully connected connections, it is proportional to its square; and the design is easily scalable. Point-to-point interconnections (PTP) scale the total bandwidth, but the O(n2) scaling in connection area is in-feasible for large designs. NoC interconnects balance bandwidth scaling and area consumption between these two options. The complexity and the costs functions associated with the increase in the number of nodes is illustrated in Fig.1.5. The figure demonstrates the complexity of non-segmented buses(NS-Bus), segmented buses(S-Bus), NoC and point-to- point connection(PTP). The NoC architecture uses layered protocols and packet switched networks which consist of on-chip routers, links, and well-defined network interfaces.

There are some differences between the nature of the conventional networking and network on chip. Routing schemes are either buffering, dropping or deflection. Dropping is used in networking. TCP uses this scheme however to avoid congestion. Each node can set the maximum number of packets that it can receive. This window is sent as part of the ack reply. In TCP

Table 1.1: Cost Function [15]

Architecture	Total Area	Power Dissipation	Operating Frequency
NS-Bus	O($n^3\sqrt{n}$)	O($n\sqrt{n}$)	O($\frac{1}{n^2}$)
S-Bus	O($n^2\sqrt{n}$)	O($n\sqrt{n}$)	O($\frac{1}{n}$)
NoC	O(n)	O(n)	O(1)
PTP	O($n^2\sqrt{n}$)	O($n\sqrt{n}$)	O($\frac{1}{n}$)

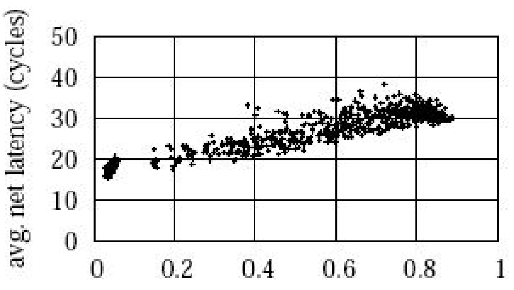


Fig. 1.6: Average Network Latency in NoC [29]

there is no way for the sender to know the maximum bandwidth. Therefore, it starts with four segments. If it receives an ack, it doubles the number of segments. When it does not receive an ack, it reduces the congestion window once again. However, buffering and deflection achieve better performance on NoC routing. Network on chip is restricted with chip area and power constraints. The allowable window of in-flight data is much smaller than in a large-scale network because buffering structures are smaller. Implementing complicated schemes on Noc would occupy area that should be dedicated to IPs. Unlike conventional networking, per-flit network latency in NoC generally remains stable even under heavy load as shown in 1.6. Therefore, latency can not be a core metric to measure congestion. The congestion in network on chip is moved from inside the network to the input queues of the network. Therefore, the congestion in network on chip is represented by the starvation rate at each node.

1.4 Network topologies

Network offers multiple concurrent links which differs for each topology. The bisection bandwidth and the diameter set the bounds for overall network performance and energy efficiency. In semiconductor manufacturing, NoCs typically favour using the two-dimensional topology like mesh. NoCs also favor flattened butterfly.
Radix denotes to the router degree. It is the number of I/O ports. The complexity and the cost of the router increases by increasing the radix; because the number of buffers and arbiters increases and the size of the switch increases. For a topology with many links, the average routing distance between two cores is small and the latency through it is low, however it is hard to layout and it consumes large area. Topologies with relatively few links are easy to layout such as meshes, tori, and trees[31].

•*Butterfly network* : has no path diversity. It is rarely used in NoCs. The hop count is constant.

•*Ring networks* : Nodes are connected in a simple circular way. Ring networks are not scalable networks. The bisection bandwidth is constant.

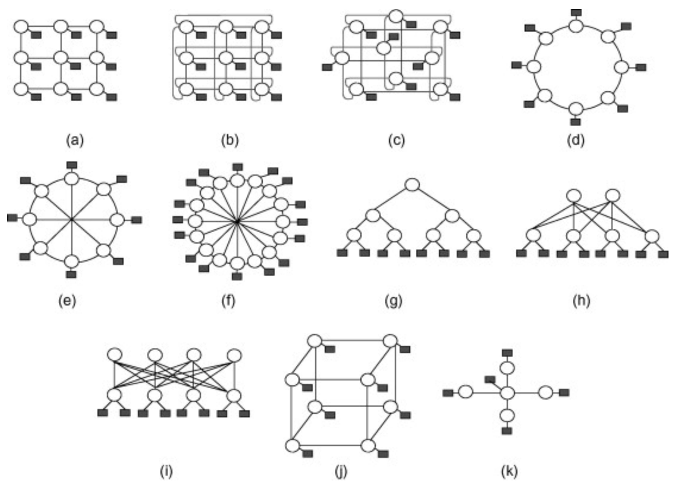


Fig. 1.7: Network Topology[18]

•*Mesh networks* : Have good path diversity and connection redundancy without being totally connected. Mesh topologies have high bisection bandwidth and sustain high traffic loads before their saturation points. Mesh topology is expected to perform better under neighbour to neighbour traffic (unbalanced traffic).

•*Octagon network* : begins routing packets along the ring using the first crossbar before using the second crossbar for either cross-connection routing or final packet transfer to the attached node.

•*Torus* : Just like mesh topology but the routers at the edge are connected to routers at the opposite edge via wrap-around channels. Torus provides higher path diversity and bisection bandwidth than mesh topology. It has higher cost than mesh topology. However, it is harder to layout on chip.

•*Folded torus* : doubles the bandwidth by wrapping leftmost routers to rightmost ones and from top component to bottom.

•*Tree* : is a hierarchical topology that begins with a node connected to more than one node (children) at the lower level. This scheme continues to k levels, where each node has m nodes. An enhanced version of tree topology is fat tree (butterfly tree).

•*Fat tree* : Each node has four children and a parent. This is replicated four times at any level of the tree. Have low hop count.

Irregular network topologies could be formed by altering connections in these regular network topologies, or they could be formed as hybrid of regular topologies. Irregular topologies are used to get better result for certain application.

Table 1.2: Topologies metrics N is the number of nodes[6]

	Ring	2D mesh nxn	2D torus nxn
Number of nodes	N	n^2	n^2
Number of links	N	2N-2n	2N
Router degree/ radix	2	4	4
Diameter	N/2	$2\sqrt{N}$	$\sqrt{N}$
Bisection bandwidth	2	$\sqrt{N}$	$2\sqrt{N}$

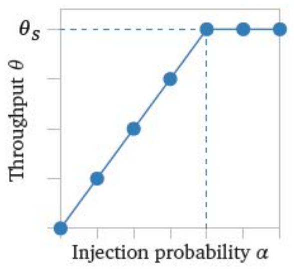


Fig. 1.8: Throughput Load Plot

1.5 Network metrics

Network is measured in terms of performance which is described as latency, throughput and path diversity. On the other side, the network architecture is restricted by area and power.
1. Latency: average delay of a packet/flit traversing the NoC. It is the packet delay represents the elapsed time from the cycle the first flit of a packet is injected into the network until the cycle its last flit is delivered. The lower bound of average latency is when the network has no packet blocking, in this case the latency is called the best case latency or zero load latency. The zero load latency is the head latency and serialization. The head latency is measured by number of hops between nodes as well as the time required to traverse a router and the corresponding link. The number of hops between two nodes depends on the chosen path and the chosen routing algorithm. The total average number of hops is determined by the diameter and size of the network; where the diameter is the highest minimum hop count. The delay of one hop depends on the complexity of the router and its critical path.

2. Throughput: is the rate of transfer of data in run time. In simple bus topology, throughput is simply the reciprocal of the latency. This is not the case in a network where the throughput depends on the path diversity between the source and destination. It is equal to a parallelism factor times the reciprocal of latency. Throughput depends on the traffic patterns. As the number of paths connecting two nodes increase, adaptive routing could be used to increase the throughput. Throughput is the measure of the maximum sustainable traffic.
The injection load is the number of flits injected in a network. The number of injected flits is equal to the number of ejected flits in the network. Once the injection rate reaches the saturation point, the network can not deliver the flits as fast as they are created anymore. Therefore, the packets will be queued in injection buffer and not injected. Thus, the throughput saturates after the saturation throughput Fig.1.8.

3. Maximum operating frequency is one of the important factors that influences the speed of message delivery. It defines the bandwidth of a channel; which is the channel bit width times the

Table 1.3: Traffic patterns s_i i-th bit of the source and d_i i-th bit of the destination [7]

Name	Pattern
Bit complement	$d_i = \bar{s}_i$
Bit reverse	$d_i = s_{b-i-1}$
Shuffle	$d_i = s_{i-1 mod b}$
Transpose	$d_i = s_{i+(b/2) mod b}$

maximum operating frequency. Bisection width is the minimum bandwidth over all bisections of the network. Bisection width influences both the power and the performance of each network topology.
The cost is measured in terms of power and area. Area is the number of look up tables occupied on the FPGA.

Load-latency curve: The latency is shown versus the throughput. The latency in cycles is shown at each throughput. The maximum throughput is the maximum injection rate. As the injection rate is the throughput until the network saturates.

In mesh network, though increasing the number of nodes decreases the maximum throughput per node, it increases the overall network throughput. The latency increases as the number of nodes increases. Selecting the number of nodes, there is a compromise between the latency and the total network throughput.[31]

The synthesis traffic used to evaluate the network mainly in this work is uniform traffic pattern. Uniform is a random spatial distribution. The destination for each source is chosen randomly. Table.1.3 illustrates different traffic patterns. Another important traffic pattern is hot spot, in which many packets from multiple sources destine the same node.

1.6 Soft NoC or Hard NoC

Designer should choose whether to implement the router using hard or soft implementation. Hard router is implemented as silicon design embedded on the FPGA. This is more area efficient and power efficient. The hard IPs also provide higher performance. Soft routers are implemented by the configurable resources in the FPGA. They are larger, slower however they are more flexible. The hard router would consume less than 1% of the silicon area. The soft network would consume less than 5% of the silicon area. The crossbar is the smallest module in an ASIC router; however, it is critical in soft implementation and it occupies 26% of the area[12]. The area occupied by each module is shown in Fig.1.9 and Fig.1.10.

1.7 Switching Techniques

Switching techniques show how the data flow through a switch in a router.

1.7.1 Circuit Switching

It means creating a dedicated permanent link between the source and destination. The merit is the ability to keep a guaranteed throughput between nodes in real time. Circuit switching also

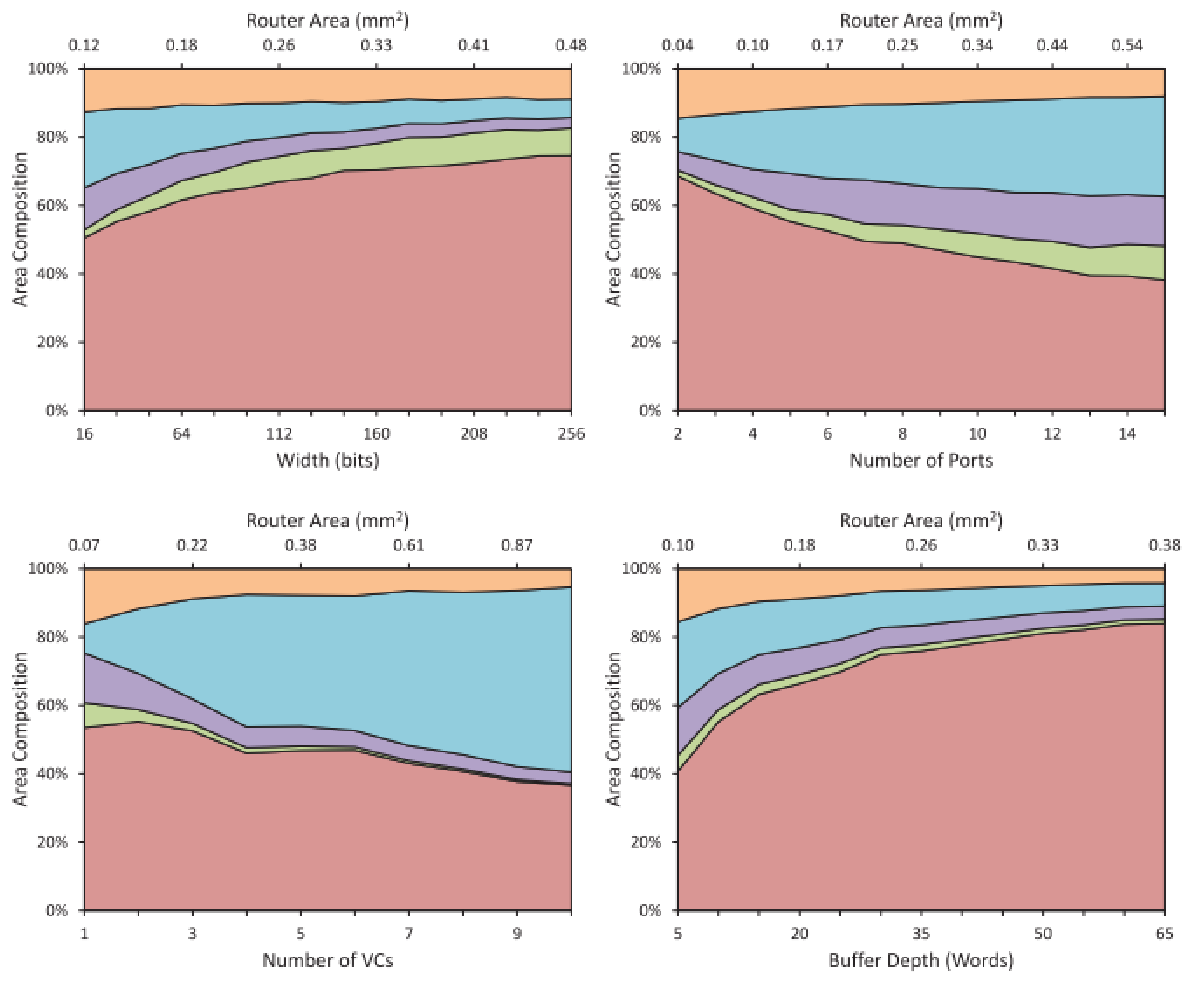


Fig. 1.9: Hard Router Area Composition Starting from the bottom: Input module crossbar, switch allocator, VC allocator, and output module[12]

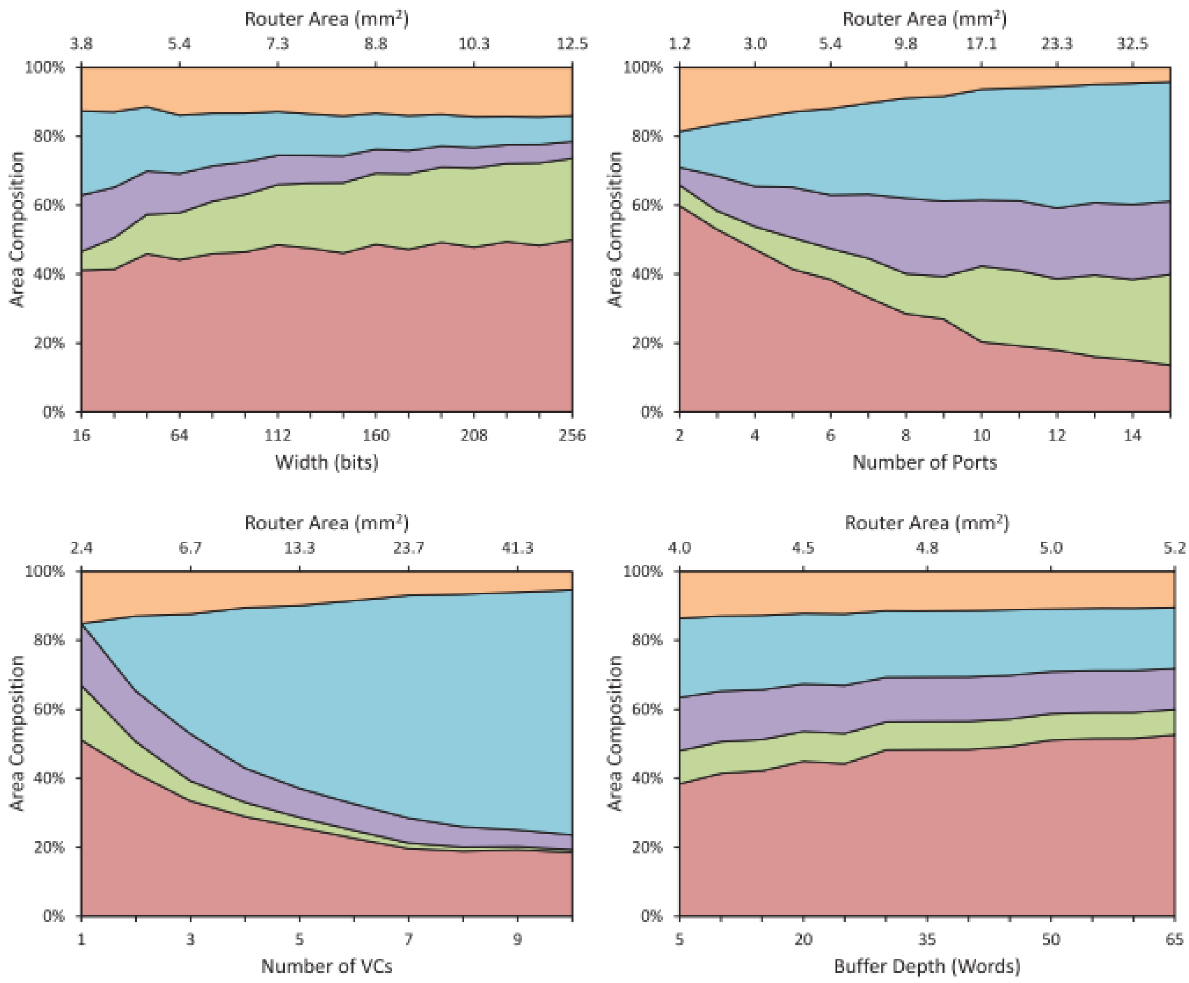


Fig. 1.10: Soft Router Area Composition Starting from the bottom: Input module crossbar, switch allocator, VC allocator, and output module[12]

minimizes the area of buffers. On the other side, resources are wasted and power is consumed when the link is idle. This happens when there is an established connection and no data transfer takes place. There is additional delay added each time a connection is being established before sending a burst of information called set up delay; also, there is additional delay added after the sending is over called tear down delay. However, the resources are allocated and maintained during data transfer, then deallocated once the last packet is received. Therefore, the streaming latency of all the data is low.

1.7.2 Packet Switching

Packet switching offers a higher throughput. Packet switching tends to route each packet independently. Channel utilization is better. The resource is deallocated once the packet moves to the next node. In circuit switching, the header only is buffered. However in packet switching, the whole packet is buffered. Packet switching improves the throughput, but occupies large buffering area. These buffers and queues are implemented as embedded memories or flip flops. Buffering schemes used in packet switching:
a. Store and forward: For the packet to be sent to the next node. The packet should be entirely received and stored by the current node. The node decodes the header after the whole packet is received. The output channel must be available and packet buffer space must be free at the next node.
b. Virtual Cut through: It reduces the latency. It is similar to the store and forward, but the packet is divided into smaller flits. The header flit can be forwarded as soon as it has been granted access to the switch, the channel, and a flit sized buffer on the next node. Transfer latency can be reduced by interpreting the header as soon as it is received. Still, Virtual cut through needs the same buffer size of store and forward.
c. Wormhole: Wormhole uses virtual channels to share the physical link between two nodes. Physical resources are allocated at flit level rather than packet level. FIFO storage is smaller than packet based storage. Packets passing through the network are divided into flits. Flit is the basic unit of the network. Each flit has the routing information. Flit can roam through the network independently. To avoid the head of line blocking (HOL) of packets in the queue, virtual channels are used. If the packet at the head of the queue can not cross the switch to its destination, another packet in the queue would take the chance. Virtual channels allow parallel queues (logical channels) to use the same physical channel. The throughput increases at the expense of the area. Virtual channels are used first with wormhole routing, but they can be used with any other routing scheme. VC requires buffer allocation for each logical channel. Multiple virtual channels represent one physical channel. Because each input port is divided into a number of queues, and each queue represents a virtual channel or a logical channel. Virtual channels improve link utilization, and therefore, it is used to improve the throughput. A large number of virtual channels may have an adverse effect on network performance due to the overhead in control logic and in multiplexing of the additional channels. ”The virtual channels could be used with applications that require traffic isolation”[45].

1.8 Flow Control

It governs how the routers communicate with each other and manages the buffer storage. It has a significant impact on the performance.

Flow control is a back pressure mechanism, to avoid buffers overflow. Therefore, Flow control prevents packet dropping. Upstream nodes should know the buffer availability of the sink node (downstream node). Flow control is performed on a per VC basis.

a. Handshaking: Flits are sent regardless of the state of the buffer capacity. Upon receipt of a flit, the sink node issues an acknowledgement(ack) or buffer-full(nack) signal to indicate successful traversal of the channel or the failure traversal of the channel. Packets may have to be sent many times in a congested network, which increases power consumption at times of high activity. This used mostly in lossy systems, where packets are allowed to be dropped and re-transmitted.

b. Credit based control: The channel sink is count at both ends of the channel. The counter represents the number of vacant slots at the corresponding port buffer. At the sink the channel count is incremented when a flit is sent and decremented when a flit is received. After the sink node sends the flit, it sends a credit signal to the upstream node (source). The source buffer count is incremented. No flits are sent unless the buffer count is non-zero. The latency to restart the channel is twice the channel latency because a credit must be sent and acted upon.

c. On and Off flow control: There are no counters; instead the sink node signals the flow to be either on or off. The off signal must be sent before the buffer is full in order to make space for the flits sent, while the off signal reaches the channel source. The sink node sends an off signal when the number of the free entries in the corresponding buffer reaches a minimum threshold. So, the upstream nodes connected to the input port stop sending flits until they receive on signal. The sink node sends on signal when the free number of entries in a certain buffer exceeds a maximum threshold.

d. Peek flow control: Routers effectively expose the occupancy information of all its buffers to its upstream sending routers. Sending routers can continuously observe the buffer availability. Peek flow control reduces storage requirements by eliminating the multiple credit counters that are normally maintained for each output and VC pair.

1.9 Routing

The routing function selects the path that a given packet must take from its source to its destination endpoint. It directly affects the average hop count, latency and the dynamic power consumed by the network. The used routing algorithm affects the degree to which load is balanced across network channels. NoCs typically use simple routing logic to decrease the cost and the critical path delay. Deterministic routing keeps a predefined path between a given source and destination independent of the current network status. XY is an easy routing algorithm and it is used for mesh and torus topologies. XY routing is deadlock free. It makes decisions based on the source and destination addresses. However the network suffers from throughput degradation as the packet injection increases.
Adaptive routing makes decisions based on source and destination as well as dynamic network indicators. This routing strategy can adapt itself to the changes in the network conditions. Adaptive routing can be based on power model which adapts routing according to power conditions in order to optimize power distribution. Routing decisions are made on a per-hop basis at each routing node. Toggled-XY algorithm is adaptive routing. It toggles between XY and YX algo-

rithms. Weighted-ordered-toggle is also an example of adaptive routing. It assigns XY or YX routes based on source-destination pairs.

a. Source routing: Source routing derives its routing decision from a pre-computed global routing table, such that all the necessary routing decisions are embedded in the header to guide a packet to the receiving node. It reduces the latency through the switch nodes. However, source routing is not considered in NoCs due to its large overhead.

b. Table based routing: Only the section of the routing table pertaining to the current node is stored in the local table at each router. It can store multiple paths for the packet as well.

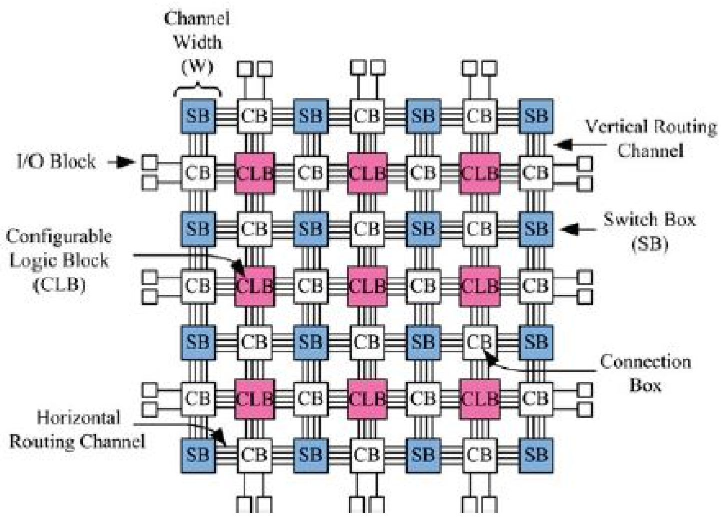
c. Algorithmic routing: Computes the output port from the destination ID. For example, in a mesh network the node needs to determine only the relative direction of the destination from the destination ID and the current node ID if absolute addressing is used. Each router is identified by its coordinates Cx and Cy. The algorithm compares router coordinates to destination coordinates Dx and Dy. When (Cx, Cy) match (Dx, Dy), the packet is transferred to local router port which means that packet reaches its destination core.

X-Y routing is algorithmic routing where a packet is first routed through the horizontal direction of the mesh until the target column is reached, then routed vertically to its destination, this routing algorithm is widely used for mesh topology, because of its low logic overhead and deadlock-free operation. X-Y routing is deadlock free. In order to avoid deadlock all cyclic dependencies must be eliminated; although it decreases the path diversity. No dependent cycles are formed, because some packet turns are not allowed. Packet turns from north-south to east-west are banned. Also, toggle X-Y routing and Odd-even are examples of algorithmic routing. Odd even prohibits the East to North-South packet turns in even columns and north-south to west turns in odd columns.

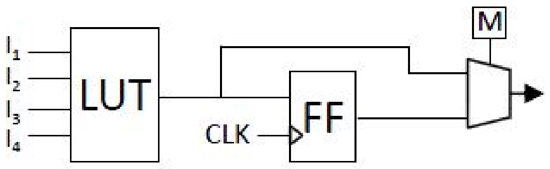
The routing logic selects the productive output port and the appropriate output VC, based on the used routing algorithm. Routing could be classified also as minimal and non-minimal. Minimal routing tends to choose the shortest available path between source and destination. Also, it could be classified as loss model and delay model. Loss model solves the contention between packets by dropping one of them, after a time-out. If the source node has not receive an acknowledgement, the dropped packet is re-transmitted. The second scheme is delay model, it solves contention between packets by buffering one of them packets at the router and tries to direct it to its productive direction later.

1.10 Unicast and multicast Routing

There are two ways of communication. Unicast which means that one node is sending packet to another node. In multicast there is one sender and more than one receiver. Multicast is used when more than one flit are destined to the same direction. They propagate through the network as one packet. The packet splits when it reaches one of its destinations, and ends when the last flit reaches its destination. Another approach to multicast routing, Each router gathers flits to form a multicast packet if they have similar direction, and split multicasted packets if their flits should diverge according to their destination bits found at the packet's header.



(a) FPGA Architecture



(b) Slice Structure

Fig. 1.11: FPGA Architecture

1.11 FPGA architecture

FPGA is divided into CLBs. Each CLB ”configurable logic block” in Virtex 7 contains two slices. Each slice has 4 lookup tables, 2 flip-flops and a multiplexer. Each 6 input LUT acts as two 5 input LUT with two outputs. This is called dual mode operation. The configurable logic blocks are connected with programmable interconnects (switch blocks). The tile is the number of CLB, switch and the area of the via in 3D. Programmable interconnects are implemented as pass transistors, multiplexers or tri-state buffer[43]. The area utilization is measured by the number of the occupied look up tables, the occupied number of flip flops and the multiplexers. However, the area utilization in ASIC is measured by the gate count. The look up table equivalence of ASIC gate counts differs from one technology to another.

To enhance the performance of the FPGA, the recent released FPGAs are either heterogenous FPGAs or 3D FPGAs. Xilinx holds the biggest share in FPGA market. According to [8], Xilinx holds 53% of the market in 2016. Altera holds 36% of the market. Xilinx offers 7 series FPGA 28nm. Figure 1.12 illustrates the difference between FPGAs in 7 series family. Xilinx also introduces the Zynq family which integrates ARM processor with 28 nm FPGA. SPARTAN 7 is the lowest priced device in 7 series family. It is equipped with MicroBlaze soft processor IP. It lacks high bandwidth transceivers. Virtex 7 offers the highest performance. In 2016 Xilinx shipped their first FinFET (16nm) 3D FPGA Virtex ULTRASCALE+. The device is equipped with up to 3.6 million logic cell. This work uses Virtex 7 as it is available and its tool is available.

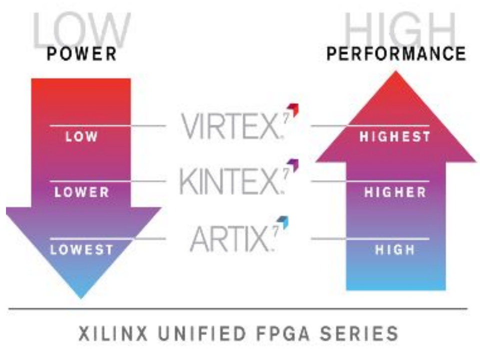


Fig. 1.12: FPGA 7 series [2]

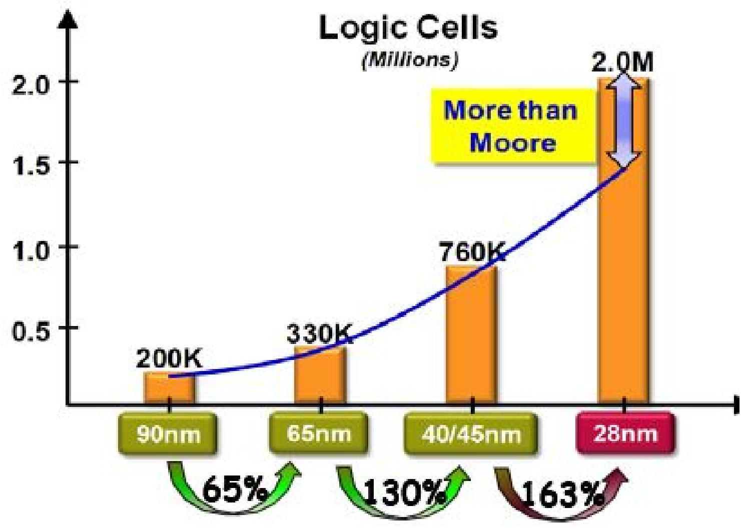


Fig. 1.13: Logic cells [2]

1.12 Power Consumption

Technology scaling reduces the power supply voltage therefore, the static power consumption decreases. Also the capacitance decreases and therefore the dynamic power decreases. However, static power consumption increases due to the transistor leakage. According to Altera report [3], as shown in Fig.1.14 at 65nm the static power overtakes the dynamic power. The main source of leakage current is the sub-threshold current.

Sub-threshold current:
When $V_{gs} < V_t$, current flows from source to drain I_{OFF}. From the equation of the current driven in [36]. There are two ways to reduce the sub-threshold current either by increasing V_t or by reducing the oxide thickness. Low threshold voltage is essential to have high speed. Also the V_t rolls off in recent technologies. This is because as the transistor channel length decreases, the drain voltage has a similar effect on the channel as the gate voltage. Therefore, less V_{gs} is needed to turn the transistor on. V_t is lower by definition. Therefore, the vertical dimensions are reduced in order to support in the reduction in the channel length. Up to a certain limit the thickness of the insulator could not be decreased anymore. Oxide breakdown is a limiting factor. Also, time dependent breakdown is also a limiting factor.

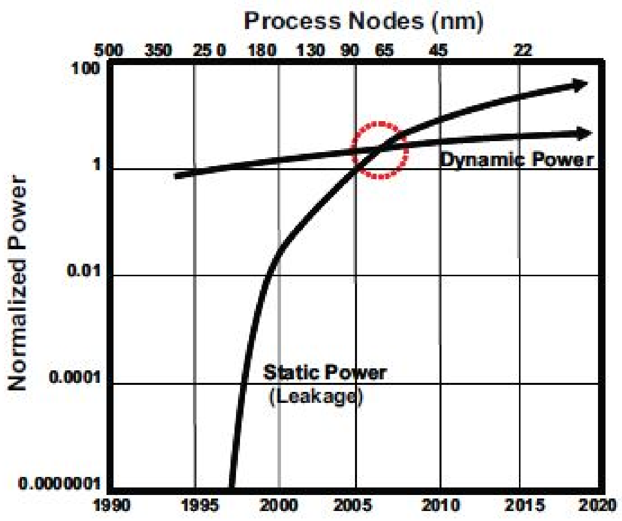


Fig. 1.14: Static power vs. Dynamic power [3]

Even at small oxide thickness, the gate would have control at the channel; but deep inside the silicon far away from the gate voltage, the drain would still have more control on the a submerged leakage path. To solve this problem, the mosfet is built on a thin silicon film. The silicon depth should not exceed quarter the gate length. Another approach to eliminate the leakage current is to provide gate control over the channel from more than one side as shown in Fig.1.15.

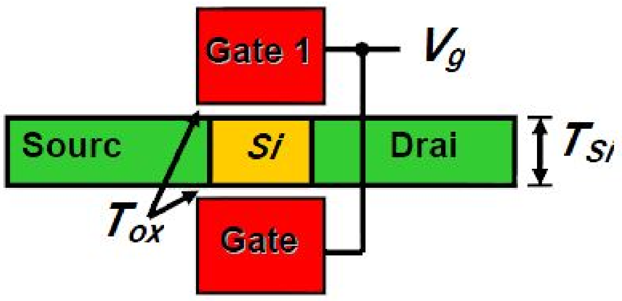


Fig. 1.15: Multi-gate Structure [37]

Gate Leakage:
The leakage gate current appears when the insulator thickness decreases. Gate leakage current increases marginally with increased temperature.

Chapter 2: Literature Survey on the Buffered NoC

2.1 Simple Router architecture

2.1.1 Buffering

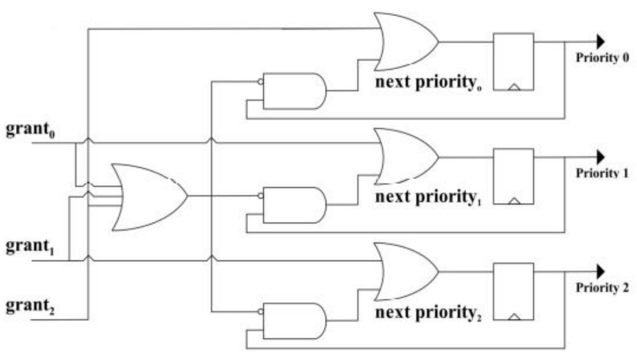
Buffering in the router on FPGA could be implemented by block RAMs(BRAMs), distributed RAMs (DRAMs) or registers. Block RAMs are SRAMs with tens of kilo bits of capacity embedded on the FPGA for storing data. BRAMs are implemented on ASIC 2D array flip-flops[34]. Distributed RAMs are small tens of bits SRAMs based on look up tables. Look up tables, used to implement functions, act as memory elements that hold the truth table of the function. Distributed RAMs are expensive as they occupy the look up tables used to implement the logic gates. Block RAMs and LUTRAMs are consumed once they are used even if only a small portion is occupied. Buffer based routers use one of these schemes. Distributed RAM is typically 1 bit wide with 16 to 64 entries. Virtex 7 has 5 address lines. It has two modes, In the first one there are two RAMs each has 5 address lines; in the second one there is only one RAM with 6 address lines. Registers make no waste in memory elements unlike BRAMs and DRAMs. Buffers could be implemented as embedded memory or flip-flops. Registers/Flip flops are rare resources on FPGA as well. Buffering is either centralized, input or output buffering scheme. Output buffers use the buffers after the allocation and switching stages; where packets are buffered in an output port of a certain router until the input port of the downstream router is ready to accept packets. Centralized buffering shares buffers between input buffers and output ones. In the routers that use virtual channels, each virtual channel is implemented by a separate buffer or queue, this to avoid head of line blocking and enhance the performance. The improper allocation of VCs can decrease the performance.

2.1.2 Arbiter

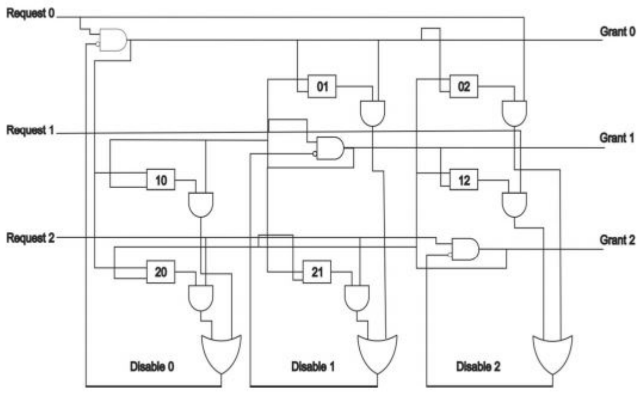
Arbiters are used to choose an agent and grant access to it in order to use a shared resource. It must ensure that only those agents that actually requested the shared resource can receive a grant. It must also ensure that in case of conflict, only a single agent can receive a grant. Arbiter can use a fixed scheme to prioritize the agents; and grant the highest priority agent out of all requesting agents. Fixed arbiter is a simple arbiter. Round robin arbiter changes the priority scheme each cycle. If the arbiter is choosing out of N agents through N cycles every user will be the highest priority user once the second in priority order once,... and so on. Finally, Matrix arbiter generates the matrix upon input and output port. The matrix arbiter sets the corresponding bit which is requested for the same output port. It provides strong fairness. It serves the agents by a least recently served policy. The precedence of each pair of inputs is saved in a register, the arbiter tracks these registers and updates it. Matrix arbiter is scalable than round robin arbiter. It grows logarithmic with the increase in the number of inputs. It occupies less area.

2.1.3 Allocator

Allocator is used to coordinate access of multiple agents to multiple resources. Separable allocator is splitting the allocator into two successive independent arbiters. For Separable input-first



(a) Round robin arbiter



(b) Matrix arbiter

Fig. 2.1: Arbiters[41]

allocator, the first arbiter guarantees that each agent chooses only a single resource to request. The second arbiter lets every shared resource selects a wining agent, if there is more than one agent requesting the same resource. For Separable output-first allocator, all agents send their requests to the first arbiter which lets every shared resource chooses a winning agent, the second arbiter guarantees that each agent will finally be granted a single resource, in case of two shared resources selected the same agent. However, the two successive stages in allocator might not work in phase, in this case agents might not get the best usage of resources.
Wave-front allocator is implemented as a square matrix, horizontal lines represent the requests and vertical lines represent the resources. The requests on the diagonal can be granted independently. “This allows the maximal matching by first granting all requests on the highest priority diagonal” [[13]]. If the resource is granted, a signal is transmitted downwards to ban any other agent from using this resource. The wavefront allocator by this connection supports a sort of sequential assignment of resources. As for a single resource ”a single column” the first agent ” the highest priority” checks the resource first, if it does not need it, the second agent checks it and so on. Therefore, the lower rows are masked by prior grants. The agents are sorted in a fixed order, to change the order of agents you can either change the cyclic connection or set the highest priority user at the first low.

2.1.4 Fault tolerance

A fault might appear in the network from manufacturing process due to the failure of links, buffers, allocator or the switch. Failure might be permanent static or dynamic faults. Static

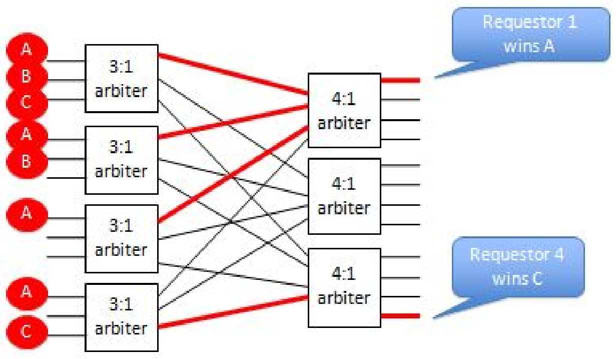


Fig. 2.2: Separable output first allocator[4]

faults like open circuit or short circuit. Dynamic faults can appear at any time. The routing algorithm should tackle this fault and keep the network running.

2.2 CONNECT

This work targets FPGA platforms in order to get use of the partial reconfiguration property of the FPGA. The user could dynamically add new routers to the design according to the density of the traffic and the complexity of the running application. Few number of designs target FPGA; most of the previous routers were ASIC oriented designs mapped on FPGA. The ASIC oriented designs does not make the full advantage of the FPGA resources. ASIC designs always achieve better performance than FPGA designs. However, if the router was designed to work on FPGA from the very beginning, it would achieve competitive performance results and occupy less area. CONNECT [45] CONNECT ”Configurable Network Creation Tool” is a network on chip generator that produce RTL specially tuned for FPGA. CONNECT offers open source verilog router code and it is specially tuned for FPGAs. Fortunately, [24] discusses adding partial configuration to the design of CONNECT mesh network. This makes CONNECT a good choice. It can have the best of the two worlds; It can achieve high performance and it can be re-configurable. The target is to create a buffer-less on-chip router that can achieve competitive performance to buffered routers. The next subsection discusses FPGA resources and how CONNECT makes the best use of it.

2.2.1 FPGA resources

a. The operation of ASIC standard cells is faster than that of look up tables in FPGA. Most of the FPGA systems tend to pipeline the design to obtain higher frequency. It is not often practical to chain up the system to large number of stages. CONNECT router is a single stage router. It makes up the FPGA low operating frequency by using wider data paths. The single stage router has added a benefit of reducing the network average latency. The design performance is better at wide data path. The single cycle helps minimizing the area occupied by the router. This is the main factor that helps reducing the hardware costs. Single stage routers also simplify the flow control.

b. FPGAs normally include:
1. Block RAMs contain tens of kilos of bits.
2. Distributed RAMs that contain tens of bits. They are implemented by filling lookup tables.

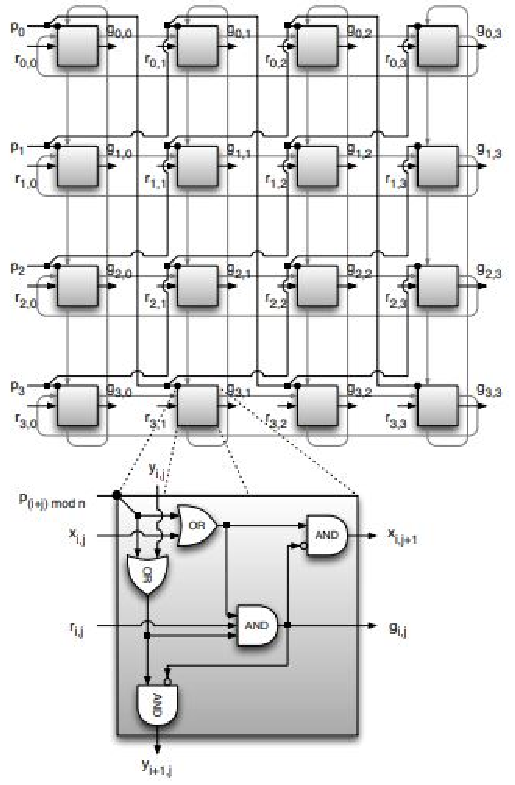


Fig. 2.3: Wavefront allocator [13]

Distributed RAMs are scarce resources as they consume lookup tables that are used to implement logic.
3. Registers are rare resources as well. Registers efficiently implement single buffers with wide datapath as mentioned in [45]. However, distributed RAMs are better in implementing buffers of large depth and narrow datapath as mentioned in [45].

CONNECT uses distributed RAMs to implement buffers and look up tables for routing algorithm. Although, routing algorithm could be replaced by routing functions especially for mesh topology, CONNECT uses lookup tables as they consume less resources than the implemented routing functions as mentioned in [45]. CONNECT implements buffers as distributed RAMs where each RAM is divided into number VCs. Each VC is implemented as a circular FIFO.

c. Reconfigurability: The design of FPGA makes it easier to remove one block and replace it with another one during the run time. This is very useful in implementing specific applications. CONNECT has multiple topologies that could be easily investigated thanks to this property.

d. Abundance of wires: CONNECT is designed to use wide datapath to make the best use of the available wires.

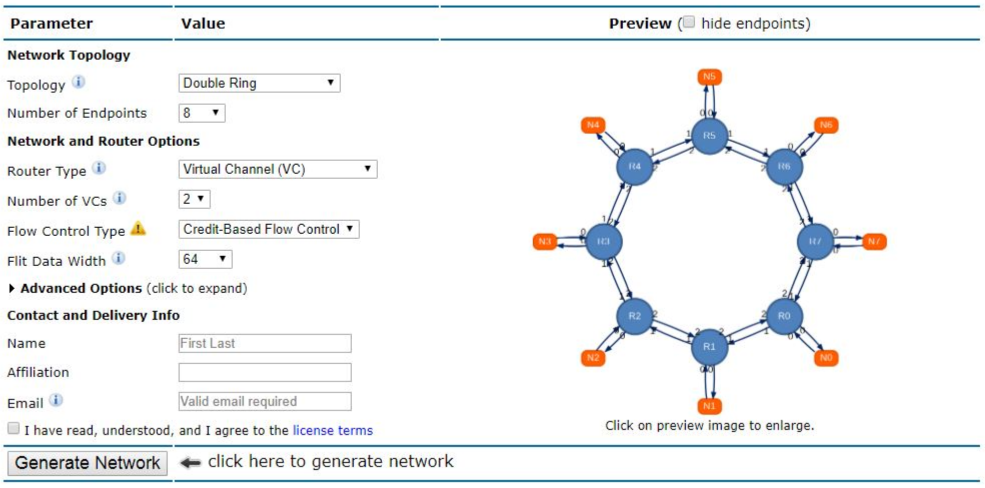
2.2.2 CONNECT generator

CONNECT [45] is a RTL generator that is based on Bluespec system verilog. The network is fully synthesizable. It is heavily configured.
The user can configure the following parameters:
1. Topology: Bidirectional: Line, ring, double ring, star, mesh, torus, fat tree, fully connected.

Unidirectional: butterfly, aggregation tree and distributed tree.
2. Number of routers: up to 128 routers.
3. Router type: simple input queued, virtual output queued, and virtual channel.
4. Number of virtual channels: up to 8.
5. Flow control: credit based control and peek flow control.
6. Flit width: up to 1024 bits.
7. Flit depth: up to 64 slots.
8. Allocator type: separable input first static, separable input round robin, separable output first and separable output round robin.

The generated code is sent to the provided mail address.

In peek flow control each router exposes the occupancy of its input buffers to its upstream routers. Sending routers observe the buffer availability. The peek flow control scheme reduces the storage requirements by eliminating the multiple counters that are normally maintained for each output and VC pair. CONNECT also single bit peek flow control which is a ON/OFF flow control.



Parameter	Value
Network Topology	
Topology	Double Ring
Number of Endpoints	8
Network and Router Options	
Router Type	Virtual Channel (VC)
Number of VCs	2
Flow Control Type	Credit-Based Flow Control
Flit Data Width	64
▸ **Advanced Options** (click to expand)	
Contact and Delivery Info	
Name	First Last
Affiliation	
Email	Valid email required

☐ I have read, understood, and I agree to the license terms

Preview (☐ hide endpoints)

Click on preview image to enlarge.

Generate Network ← click here to generate network

Fig. 2.4: CONNECT Generators [45]

2.2.3 Router architecture

This study is interested in CONNECT with this configuration.
Flit width: 64 bit wide.
Router depth : 8 slots.
Router type: simple input queued. This router has only one virtual channel per each physical channel.
Allocator: static input first allocator.
Mesh topology: 4x4 mesh.
Flow control: credit based flow control.

Basically the incoming flits access the routing tables at the routing logic to fetch their output ports. The incoming flits are registered at (input buffers) corresponding to each input port also the tags of their output ports are registered at (output module). The incoming flits use their valid bits and the output ports to arbitrate for their output ports at the (allocation logic). Then, the flits traverse the switch to their granted output ports.

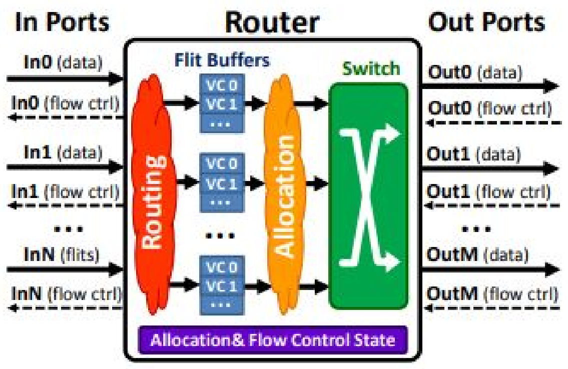


Fig. 2.5: CONNECT Router [45]

Routing

The packet is divided into number of flits. The flit is the basic building unit of routing in a network. The flit consists of data, destination, VC number, tail bit and valid bit. Incoming flits at each input port are processed with input buffers as shown in Fig. 4.16. The logic of input buffering fetches the destination bits in the incoming flit. Routing logic is look up tables that store the corresponding output port for each possible destination. Routing logic appends the output port to the flit. Look up tables are implemented as DRAMs in CONNECT. For Mesh and torus networks these look up tables could be replaced by XY routing function. However, routing tables occupy less area. They exploit FPGA resources efficiently.

Buffering

The Incoming flits are registered in the input buffers at each input port and the corresponding virtual channel. The appended output ports are registered in a circular FIFO at the corresponding input port and the corresponding virtual channel. This FIFO is called (output port module). The flit at the head of the FIFO requests its desired output port at the allocation logic. However, flits never propagate down the router logic, flits are saved in the input buffers. Number of VCs does not affect the output port module. The number of output port modules depends on the number of VCs [34]. Input buffers are implemented as DRAMS in CONNECT. In almost all the possible cases of CONNECT distributed RAM occupies less than 10 LUTS. Using virtual channels increases the performance of the router. Each virtual channel or logic channel is implemented as a circular FIFO. The distributed RAM is split into several fixed regions. The head and tail for each FIFO are implemented as discrete registers. This design allows the number of VCs to scale easily. This makes CONNECT a flexible design. CONNECT offers a feature ”virtual link”. This feature allows the packet to be sent continuously. It guarantees that when a packet starts sending its flits, it will not be interrupted by another packet. Therefore, flits can reach the destination in order. There is no need to use assembly buffers at the nodes.

Allocation

To determine which flit departs the router flit header, buffer occupancy of the corresponding input port and the credit availability are forwarded to the allocation logic to arbitrate for the output port. CONNECT supports four different allocators. Discussing the separable input first allocator as an example, the allocator has two arbiters. Input arbiter where each incoming flit from each input port requests its desired output port. Each input chooses its most appropriate output port according to a fixed priority scheme. The output of the input arbiter is fed to the output arbiter. The output arbiter allows each output port to choose the input port. Each output port prioritizes input ports in a fixed order. So that if more than one input port request the same output port, only one input port is granted the permission to use this output port. The arbitration logic respects the port priorities and VC priorities.

Switch

The switch allows the flit registered at the input buffer to pass through the granted output port. Switch output triggers the input buffer to shift and to receive new data. Switch output triggers five multiplexers to direct the flit to its granted output port. The switch is implemented as an encoder.

CONNECT uses two channels. One for data and the other for control(backpressure bits). Both channels are bidirectional.

2.2.4 Simulations

CONNECT is evaluated against network on chip designed for ASIC. CONNECT achieves better performance than SOTA while reducing logic utilization. CONNECT routers use between 40% and 50% fewer LUTs. When both CONNECT and SOTA operate at the same frequency 100MHz, CONNECT achieves lower latency and better saturation throughput with less than half area utilization. CONNECT achieves three to four times saturation throughput of SOTA. On the other hand when each router operates at its maximum frequency, SOTA outperforms CONNECT because of its higher operating frequency. Different network topologies are simulated to demonstrate the flexibility of CONNECT. The performance depends on network topology and configuration; it depends on the traffic as well. This demonstrates that designs implemented especially for FPGA can take advantage of FPGA resources and obtain comparable results.

2.2.5 Parameters

Increasing the buffer depth, number of VCs, the number of output ports or the flit width have negative impact on the frequency. Distributed RAMs are used to implement look up tables that save the output port. It is recommended to use BRAMs at input buffers instead of DRAMs to speed up the router. Block RAMs are totally consumed if only small portion of it is used. These recommendations target saving the performance and area utilization. If the reduction of power is the target, it is recommended to use DRAMs in implementing input buffers instead of BRAMs. BRAMs consume more power than DRAMs. BRAM gives the best area utilization. DRAM starts to occupy more area as the data width becomes wider. DRAM starts to occupy more area also as the buffer depth increases or as the number of VCs increases. Increasing the buffer depth have strong impact on the area of the distributed RAM. Wider area resources scale smoothly, while taller memory arrays scale in an abrupt step-wise manner.[45] Increasing the number of VCs makes the BRAMs more efficient. If the buffers are replaced with a single pipeline register, the pipeline register is better implemented as a register. Increasing the data width of the flit reduces the area utilization. Number of input buffers equals to number of input ports.
Number of encoders (switches) is equal to the number of input ports. The encoder is purely combinational logic, therefore the area occupied by the switch on FPGA NoC is larger than that

in ASIC.
There is only one allocator. The number of inputs to the allocator is equal to the number of input ports times number of output ports. The allocator consists of two arbiters, the static output arbiter consists of a group of multiplexers. The number of multiplexers is equal to the number of output ports. Each multiplexer has inputs equal to the number of input ports.
Number of output modules is equal to the number of input ports times the number of virtual channels. The depth of the output port FIFO is equal to the depth of the input buffer. The width of the output module is equal to the number of bits needed to represent the number of output ports of the router. The Number of lookup tables increases as the number of input ports increases and as the number of output ports increases.

From this discussion, it is clear that the number of input ports has the largest impact on the hardware cost followed by the number of VCs. Changes in the buffer depth or the flit width have a lower impact, basically they influence the buffer resources only.

A study is conducted to compare two software configurations with hard implemented CONNECT [23]. In each configuration area, power and speed are measured. The first soft implemented targets the speed and the second configuration targets the power. In the study the FPGA synthesis, mapping and place and routing are changed to decrease the gap between soft and hard implementations. For example DRAMs are used in power configurations while BRAMs are used in speed configurations. In the configuration that targets the speed, the minimum delay gap between ASIC and FPGA is 5.5, power gap 12.2 and area gap 5.9. In the second configuration that targets the power, the minimum delay gap is 6.3, power gap is 4.5 and area gap is 6.9. The mentioned gaps are the ratio between the geometric means of each of delay, power and area on FPGA and the corresponding value on ASIC.

Chapter 3: Comparative Study of Buffer-less NoC

Network designs assume that each router needs to contain a buffer to hold flits. Actually, buffering the flits improves the throughput. However, buffers are eliminated to decrease the area utilization of the network. Each node has the injection buffer; there are no buffers aside from the pipeline registers. Although buffers give large bandwidth. They consume huge area. Any buffered routing needs a control flow or some sort of communication between routers to avoid buffer overflow. In buffer-less systems there is no need to any control flow, as there are no buffers; and any link is available at each clock cycle. So, it is proposed to eliminate the buffers.

There are two approaches to create a buffer-less system. Either Drop or Deflection routing. When two flits contend for the same output port, the router drops one of them. If the timeout is over and the source has not received an ack from the destination. It sends the packet again. Deflection routing achieves better performance than the dropping approach.

When two flits contend for the same output port, one is directed to its productive direction and the other is deflected to any other available output port. Deflections spread traffic away from hotspots and balance load in unbalanced traffic patterns. It is a sort of adaptive routing.

Buffer-less router trades performance for low area and less power overhead. Buffer-less routers can obtain 40% savings of buffer-less router power [44]. For Routers on FPGA around 30-40% of their area resources are consumed by the crossbar and 20-40% by the buffering logic [40]. Flits are temporarily held in pipeline registers within each router and between each router pipeline stage, until an output port is allocated. If the assigned output port is not the desired one, it is known as a deflection. To apply buffer-less algorithm, the number of output ports should be equal to the number of input ports. The router should be reachable from other routers. The common node injects as long as there is at least one free output port. Buffer-less networks provides backpressure only at the local injection queue. Allocation logic in buffer-less designs is slightly more complex than that in buffered designs. As input flits must be ordered and prioritized to avoid live lock problem. Flits with higher priority are served first and they are granted their requested output ports. Allocation logic should guarantee that low priority flits are deflected to any output port. At each clock cycle, the router should be ready to receive new input flits at each input port, if the router is a single stage pipeline. Generally, the flits that arrive in a given cycle can always leave exactly N cycles later. Each buffer-less architecture should address explicitly how to solve live-lock. As flits might roam continually through the network, they might never reach the destination.

Buffer-less designs work efficiently at low traffic loads. However, they have lower saturation than buffered designs. Buffer-less system can not sustain high real time traffic. Both the latency and the throughput degrade as the traffic increases. Many designs have been proposed to improve the performance at high injection rates. Another main disadvantage for buffer-less router designs is that buffer-less routing can not use pure wormhole routing algorithm; because, packets are divided into flits, each flit travels through the network independently. When two flits contend for one output port in a router, the buffer-less router avoids the need to buffer by misrouting one flit to another port. The flits travel through the network until ejected at their destinations, possibly out of order. Therefore, packet's flits might reach the destination out of order. This is why assembly buffers are used at the nodes or IPs.

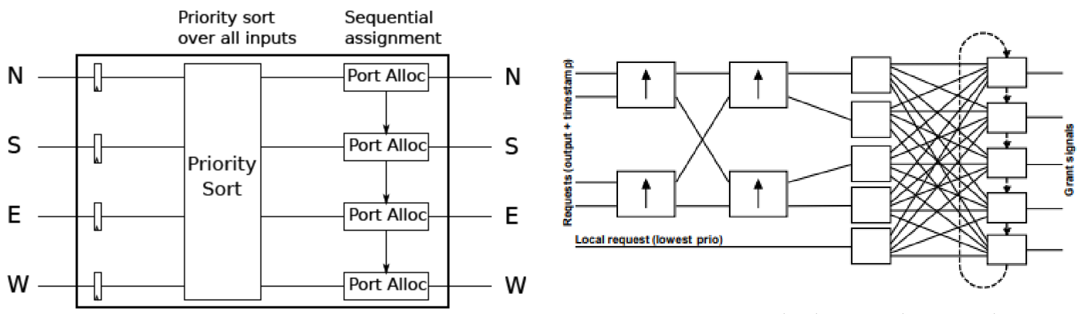
Reassembly buffers: The buffer-less routers need assembly buffers at each node. Packets are divided into flits. The header information is attached to each flit. Each flit is routed independently. Each flit may follow different path. Therefore, flits of the same packet do not reach their destination in order. The node receives the flits interleaved and each node needs buffers to re-assemble the flits of the same packet. Small assembly buffers could cause deadlock. Flits might be dropped because of these assembly buffers; as there is no backpressure for these buffers. In Chipper [20] the source router waits for acknowledgement from the sink node. If the source node haven't received an ack and the timeout has passed, the source node re-transmit the flit again. However, there is a separate network links for these acks transmitted between sources and receivers to guarantee that the flit is not dropped.

Mesh topology is preferred in implementing buffer-less network; because it offers parallelism. Generally, any topology that is used in buffer-less NoC should satisfy these two conditions. First, the number of input ports in every router is equal to the number of output ports, this is essential to prevent flit drop. Second, Every router is reachable from every other router.

A brief overview on different router architectures.

3.1 BLESS Architecture

BLESS [44] is a three stage pipeline buffer-less router. The router uses age priority. Incoming flits are kept in pipeline registers till output ports are allocated. Requests are fully sorted according to their age by sorting blocks, each of which sorts two requests as shown in Fig.4.10. A time stamp is added to each flit. The allocator assigns each input flit an output port, where the oldest flit is guaranteed not to be deflected in order to avoid live-lock. The allocator is divided into simple arbiters as shown in Fig.4.10 where the oldest flit is granted its requested output port, and the conflicting flit is forwarded to the next arbiter to be deflected to another output port. Each small output port arbiter should wait for the previous output port arbiter. This sequential logic creates a long critical path; the critical path scales with the number of input and output ports. Critical path degrades the network frequency and increases the average network latency. BLESS has the highest throughput in terms of cycles and the highest saturation out of the currently proposed buffer-less routers. This router guarantees that the maximum number of flits will get their requested output port; however in case of contention some flits are obligated to be deflected away from productive direction. BLESS is a three stage buffer-less router. Here is the description of the router pipeline. First, the router computes the requested port of each incoming flit. The route computation logic computes the requested output port based on XY routing algorithm. The router compares its index with the destination index. The router picks the x direction over the y direction. Then, the incoming flits are directed to the routing logic that sorts the flits and grants output ports. The flits must be sorted before arbitration. Later, incoming flits are directed to the switch (crossbar) which consists of multiplexers that direct the incoming flits to the granted output ports. The header and the payload of each flit, both travel through the network. The packet is divided into flits. Buffers and router resources are assigned per flit basis. The flit is the building block of the network; the header of each flit contains the age, the destination, the source and a valid flag. There is no pure wormhole switching; each flit is routed independently. Age priority prevents live-lock. The router occupies large area, because it has many comparators, as the router makes full sorting to the incoming flits by the age priority. Age priority is expensive in header information and the critical path circuit. Though the operating



(a) BLESS Allocator

(b) BLESS Priority Sorting Logic

Fig. 3.1: BLESS Architecture [44]

frequency is low, the router is used as a basis router to buffer-less routers in simulations. Also, the network saturation throughput of BLESS is less than that of buffered routers. The average packet latency increases because of the deflections and the long critical path. BLESS can cope with the bursty traffic as the network is self-throttling. Deflection algorithm behaves similar to adaptive routing algorithms as it avoids congested paths and deflects flits towards other parts of the network. The router can not inject any new flits unless there is a free channel. The router takes the injection decision independently. It is purely local. There is significant energy saving at small performance loss. In deflection routing the body flit might not follow the head flit; as every router is routed independently. In order to use wormhole routing, every router stores the header information of all worms in transit as well as their allocated output ports to allocate the body flits. The router assumes that the reassembly buffers are infinite. The author proposes a reduced latency BLESS. Reduced latency BLESS is two stage pipeline router.

3.2 Chipper Architecture

Chipper [20] is a three stage buffer-less router. It solves the basic drawbacks of BLESS router. The main contribution is a cheap deflection router. Chipper replaces the sequential allocator with a parallel permutation network.

3.2.1 Permutation network

Sequential allocator is not essential for ensuring mutual exclusion on output ports. Parallel permutation network occupies less area than the sequential allocator logic also the permutation network deals with flits in parallel way. It has smaller critical path. Chipper can work with high frequencies. Permutation network can send a flit on any input to any output. It can give a 1-to-1 mapping of inputs to outputs. However, it cannot perform all possible permutations of inputs and outputs. The proposed permutation network is 2-stage network. Every router in the network even those at the edge of the network should have all their output ports available. For the routers at the edge of the network, the output port is also the input for the same router as illustrated in Fig.3.2. This is a key insight that significantly decreases the number of stages of the permutation network from three stages to two stages. The network sorts the input flits then allocates the ports. The permutation network is composed of 2x2 arbiter blocks that either pass or swap their arguments. Each arbiter block has two inputs, arbiter logic must allocate ports in priority order. The arbiter sorts the two flits then pass or swap. The arrow direction in each

2x2 module indicates the sort direction (increasing or decreasing). The steering function in the arbiter block is simply done by a 2x2 multiplexer to allow the block to either pass or swap. If the winner flit requests the corresponding output port, the steering function pass the input flits to output directions. If the winner flit requests the opposite direction, the steering function swaps the input then directs it to the output directions. Pass means passing the first input to the first output and the second input to the second output. Swap refers to passing the first input to the second output and the second input to the first output. The steering function depends on the requested output ports of the winner flit only. The other flit if it exists takes the remaining port. Ruleset 1 describes how the arbiter logic sorts the two flits and determines the winning flit. The permutation network does not perform full sorting of the incoming flits. Chipper relaxes the constraint and sorts the incoming flits partially. It guarantees that only the highest priority flit is granted an output port in its productive direction. This constraint is sufficient to make the network live-lock free. However, the arrangement of the I/O ports of the permutation network highly influences the performance.

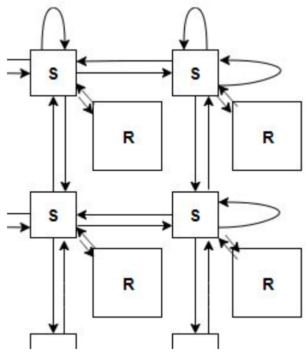


Fig. 3.2: Edge Routers in Chipper NoC

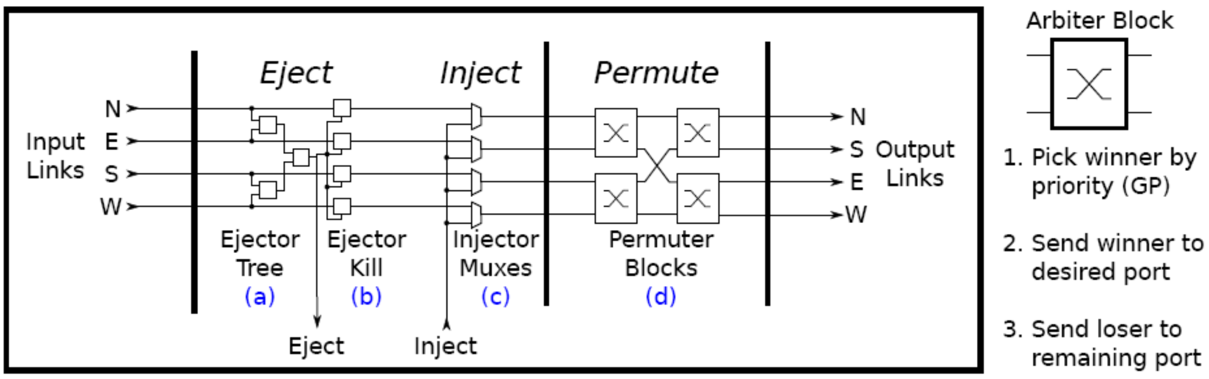


Fig. 3.3: Chipper permutation network [20]

3.2.2 Golden Flit

Chipper also uses a cheap priority scheme. Instead of using the age comparison as used in BLESS, Chipper uses a golden flit. Where for certain period of time, a single flit is prioritized over all flits in the network, this period is long enough to ensure its delivery. This assignment rotates through all possible packet IDs. Therefore any stuck flit will eventually be golden. The

Ruleset 1 Golden Packet Prioritization Rules

Golden Tie: If two flits are golden, the lower-numbered flit (first in the golden packet) wins.
Golden Dominance: If one flit is golden, it wins over any non-golden flit.
Common Case: Contests between two non-golden flits are decided pseudo-randomly.

golden flit wins the arbitration over all the common flits. The logic ,which calculates the golden epoch and determines the golden flit, is either a central logic or distributed logic. This work uses a pseudo random function at each router to calculate the golden flit. This work uses distributed logic. This simple priority scheme guarantees that the network is live-lock free.

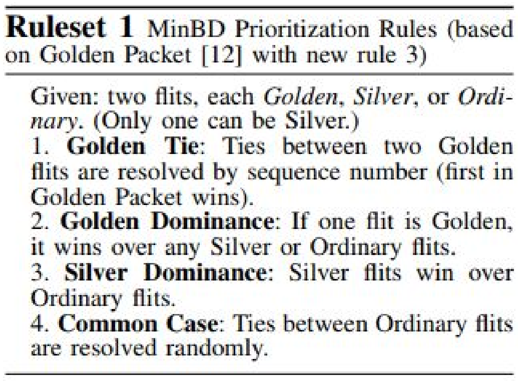
The router is three-stage router. The longest path is the permutation path. Therefore, eject and inject units do not affect the frequency. After Eject and Inject units, the route computation logic is used to determine the requested port of each flit. The router determines the golden-status of a packet in parallel with route computation.

As Chipper guarantees only one flit to be directed to its requested output port, the throughput of the network is low compared to BLESS and buffered architecture. Chipper has high deflection rate. It saturates faster than BLESS. Chipper can work on high frequencies, therefore the overall performance of Chipper is better than BLESS. Chipper reduces the average network power by 54.9% in a 64-node system compared to a conventional system. It reduces the power by 8% compared to BLESS. Chipper is the only router that addresses the assembly buffers problem. The flits do not reach the destination in order because of the deflections. Chipper uses cache registers as assembly buffers at each node.

3.3 MinBD Architecture

MinBD [21] is a 2-stage buffer-less router. MinBD only enhances the throughput of Chipper. Chipper performance degrades at high traffic. The rate of deflected flits increases at high injection rates, therefore the performance degrades and the dynamic power increases. Therefore, MinBD proposes using a side buffer. The side buffer is a small buffer that keeps flits rather than deflecting it. The router examines the flits at the output of the permutation network, if the flits are deflected the router chooses one of the deflected flits and saves it in the side buffer. It is mentioned in the paper, that the presence of the side buffer actually decreases the throughput, as the increased throughput places more ejection pressure on the nodes, so it should be used with dual eject unit. Therefore, MinBD increases the throughput of Chipper and it increases the area only by 3%. Also, the critical path does not increase, as the longest path is still in the permutation network. Adding dual ejection to the side buffered system to address the ejection bottleneck increases performance to 5.8% above baseline Chipper.

The presence of the size buffer is more important than its size; as its utilization is low. The injection of flits from the side buffer has higher priority than the injection of new traffic. To avoid starvation of flits inside the side buffer, a threshold is determined. If the time exceeds this threshold, the flit is injected and another is placed in the side buffer. The golden epoch time is equal to the threshold of the side buffer. The buffer depth is chosen to be 4 flits. As the size of the buffer increases, the consumed power increases without any significant improvement in the performance. MinBD increases the critical path relative to Chipper by adding the redirection and re-injection logic.



Ruleset 1 MinBD Prioritization Rules (based on Golden Packet [12] with new rule 3)

Given: two flits, each *Golden*, *Silver*, or *Ordinary*. (Only one can be Silver.)
1. **Golden Tie**: Ties between two Golden flits are resolved by sequence number (first in Golden Packet wins).
2. **Golden Dominance**: If one flit is Golden, it wins over any Silver or Ordinary flits.
3. **Silver Dominance**: Silver flits win over Ordinary flits.
4. **Common Case**: Ties between Ordinary flits are resolved randomly.

Fig. 3.4: MinBD Flit Sorting Algorithm [21]

The second contribution is the silver flit. When no flits are golden, unnecessary deflections take place because the permutation network stages are not coordinated. A flit might win the arbitration at the first stage and let another flit be deflected and then loses the arbitration at the second stage and also be deflected. The router adds the silver flit to ensure that at least a flit will certainly not be deflected when there is no golden flits. The silver priority level wins arbitration against common cases and lose to the golden flits. It guarantees that at least one flit is prioritized over all flits in every router. After adding the silver flit, the sorting is done as illustrated in ruleset1.

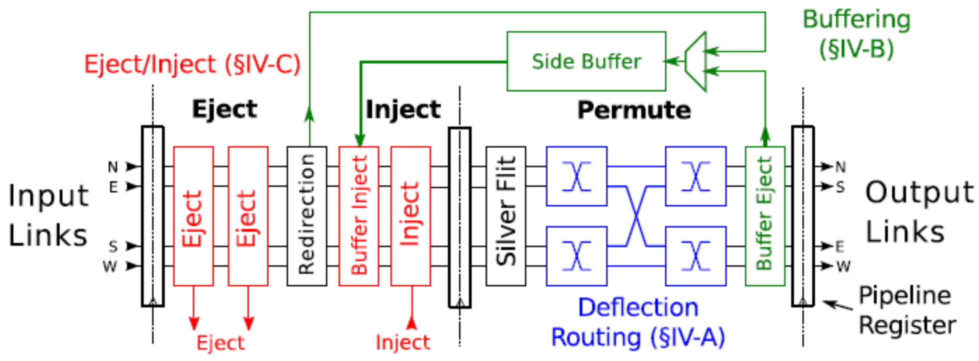


Fig. 3.5: MinBD router architecture [21]

3.4 SCEPTER

Scepter [17] is a 3-stage buffer-less router.

3.4.1 SMART INTERCONNECT

Scepter tries to make latency independent of hop count. The flit sends an arbitration request to more than one router along the path of the flit. This look ahead arbitration allows the flit to make multiple hops in one cycle. It allows the flit to cross multiple routers in one cycle. This idea was used before by SMART buffered router. And it is useful to be used in deflection routing. A smart hop setup request is sent a cycle in advance along the path to destination to arbitrate for the crossbar switches. There is a maximum number of hops that could be covered in one cycle. The SSRs travel through dedicated wires. A flag is saved in each router the smart hop setup request (SSR) passes by to indicate whether the arbitration succeeded or not. The flit does not need to wait for a grant signal to indicate the arbitration success. The router prioritizes flits from neighbourhood over far requests. SSR requests are prioritized using a fixed scheme. SSR requests are prioritized by the distance to the destination.

3.4.2 Routing Logic

At the end of first arbitration stage, the local flits are assigned output ports. The highest priority flit picks the output port first. If the assigned output port is in progressive direction, SSRs are sent along the path to the destination. The second arbitration stage arbitrates between SSRs, local injected flits and flits from the neighbourhood. In the common case, flits nearer to the destination are prioritized in order to be drained out of the network faster. This priority scheme is used to make the average network latency short. The sorting of SSR depends on the nearer to the destination. The destination proximity scheme is used to drain the flits quickly out of the network and reduce the average latency of the network. To avoid starvation of injection when the injection queue depth is larger than certain number of entries, priority order changes to prioritize injection over far SSRs. The maximum number of inputs is four whether SSRs or flits from neighbouring routers. Therefore, if an incoming flit and a SSR come from the same input direction, the router prioritizes the neighbouring flit over the SSRs and SSR over local injected flits. To arbitrate between neighbouring flits, flits closer to the destination are prioritized over far flits. The flit chooses the shortest path to destination. Therefore, it toggles between XY and YX directions continuously. The flit chooses the minimal path as long as it is not deflected more than once. The network toggles between XY and YX priority scheme to avoid ping pong effect and dynamic power loss. This is applied when the flit is deflected more than once(ping pong), the router increases the priority of the flit and directs it to another output port.

3.4.3 The Highest priority source ID

To avoid live-lock problem a synchronized time window at each node in the network is calculated, and a consistent highest priority source ID at each time window is enforced. ”Flits in the NoC that originate from this source are prioritized over others. The highest priority source ID rotates each time window, where the time window length is sufficient to drain the network of a request from this source.” [17]

3.4.4 Throttling techniques

Adjusting the global throttle rate does not relieve the congestion at all the nodes in the network especially the nodes at the center. Therefore, using a distributed throttling is more effective as each node can rely on both the global throttle flag and also on its experience. A starvation flag is set when the number of flits in the injection queue exceeds a certain threshold. To avoid starvation of the nodes in the network. The most straightforward is ON/OFF throttling. It stops the non-starved nodes from injection and allows the starved nodes to inject. SCEPTER uses Q learning algorithm. Q learning helps the node to take the best action for an environmental state. For each node, there is a table of states and actions. State is the state of starvation for the local node and across the network. Every L cycles a global starvation flag is sent. There are 8 states for the network. Action is whether to increase or decrease the throttle rate. There are 5 possible

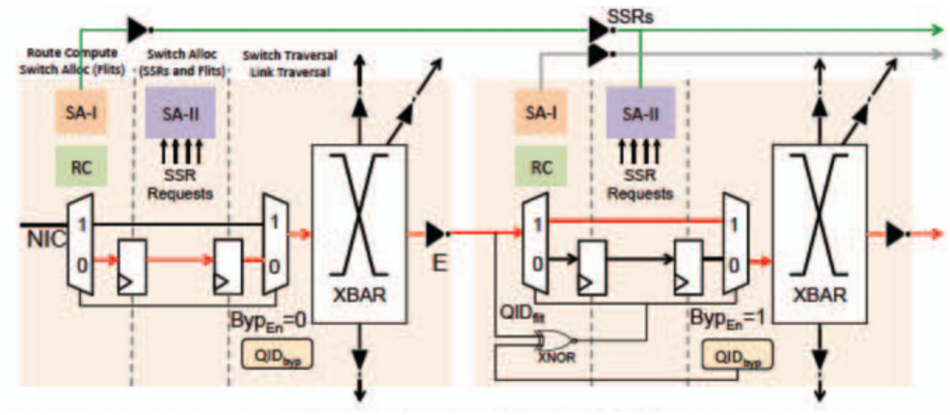


Fig. 3.6: Scepter router architecture [17]

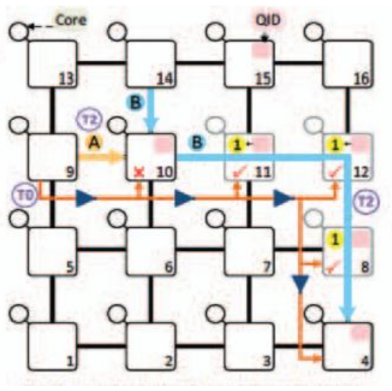


Fig. 3.7: Opportunistic Bypassing [17]

actions. The 5 actions are to increase, decrease or to retain the throttling rate. There are two degrees to increase, either to increase slowly with small step or to increase largely with a large step. The state of the node is calculated according to a throttling equation and Q table is checked. The action that has the maximum Q value is chosen. The throttling rate converges according to the equation. The throttling equation is function of the learning rate which indicates ”how quickly new congestion information is factored in update” [17].

3.4.5 Opportunistic Bypassing

A flit can take advantage of a pre-reserved path to another flit that could not arrive at this clock cycle. If both flits destine the same direction. Fortunately, this flit could make multi hops in a single cycle taking advantage of a path reserved by SSR of another flit. At T0, Flit A sends SSR from node 9 to 4. Another flit B reaches 10. SSR could reserve the whole path for flit A except router 10. At T1, flit A reaches router 10. Flit B uses the path and reaches router 4.

SCEPTER is compared to BLESS and a buffered router. It reduces the average latency by 62% compared to BLESS. It has 1.3 higher saturation throughput. SSRs are 10% of switch allocation. Half of them is wasted. In the other half a flit actually traverses the crossbar. Self throttling algorithm affects the large network diameter more than the small network. It reduces the starvation rate of 256 nodes by 38%. It reduces the average latency by 24%. The self throttling is used to achieve equal bandwidth at network nodes by calculating the per-node throughput. ”ON and OFF throttling does not provide any fairness improvement.”[17] SCEPTER occupies less area than buffered baselines. SCEPTER occupies 16% area more than BLESS and 13%

power higher than BLESS. SSR creates dynamically multi hop paths. The router ensures high throughput and fairness.

3.5 Carpool

This buffer-less router supports multicast forking and hot spot merging. Carpool [30] is a modified version of NOCulator, an open source network on chip simulator.
Carpool merges flits sent by multiple nodes to a common destination node. The intermediate routers between source and destination can decide to fork new replicas of the flit only if there is available output port for each copy of the flit to avoid dead lock problem. The replicas of the flit are produced based on the destination list in the header of the multicast flit. The intermediate routers between source and destination can also merge multiple flits into a single flit if they are on their route to the same destination.

3.5.1 Forking Flits

Injecting a multi-cast flit with m destinations can take m cycles. Because there is no hardware to support m parallel injection. There must be output port for every copy to fork a multi-cast flit. So generally, this equation must be satisfied.
The incoming flits - removed flits (either ejected or merged)+ replicas should be less than or equal the number of output ports.
The forking of multi-cast flits increases the number of flits in the network. Forking is suitable for low load applications. It decreases the network latency and improves the frequency. However, at congestion the forking is disabled. As Forking increases the number of flits in the network, and this may decrease the saturation throughput. If the network is congested, the router disables the forking property. If the multi-cast flit is not forked, it travels to its destinations sequentially. If a multi-cast flit passes by a router that has disabled the forking, a single port is allocated to the multi-cast flit and the flit travels sequentially to each of its destinations. A copy of the flit is ejected and another continues to the next destination. The flit is copied at each destination except the last one. The congestion is measured by the starvation rate. Starvation rate is the measured by a counter in each router to count the number of cycles that the router could not inject any flits in the network.
To avoid the high network latency, the re established path can not be used. The destination list becomes larger when merging multiple flits. To avoid scaling up of the header bits, the network is divided into clusters, the destination list is represented by clusterID, destID. The header bits have cbits to encode the cluster and m bits to encode multiple routers. The destination bits are c+m bits. A single multi-cast flit can destine m routers inside the same cluster. The destination routers must be in the same cluster. Carpool uses fewer flits in sending multi-cast than using unicast flits as traditional buffer-less networks. The number of flits in a single multi-cast flit sent to d nodes is b(h-m), where b is the size of the request and h is the data size. In traditional buffer-less network the number of flits sent to d nodes is d*(b/h).

3.5.2 Merging Flits

In order to merge two flits the router should check that the two flits have the same payload and the same destination and come from the same cluster. When a flit is at certain input port, the

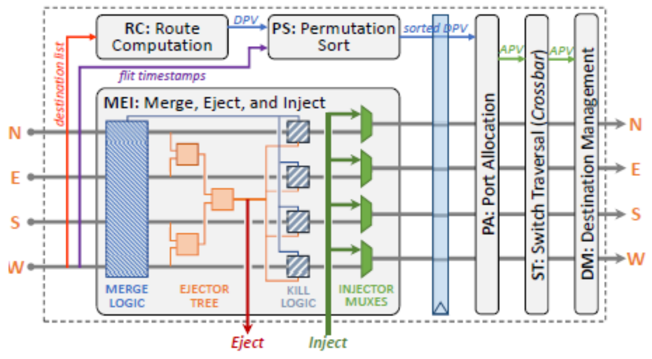


Fig. 3.8: Carpool router architecture [30]

router checks the flits at all the higher input ports searching for a flit to merge with. For hotspot request when an intermediate router detects two requests destine the same node and have the same payload, the router merges them into one request. The router uses the same encoding to encode the source nodes used with destinations in flit forking.

3.5.3 Router Architecture

First, the router merges any flits could be merged. The merge logic is followed by the eject unit. It ejects up to one flit. Followed by the inject unit, if there is an available channel, the local node injects flits. Then the router computes the route. Carpool performs XY routing to determine the desired output port. Then the router sorts the incoming flits. It arranges the flits using the age, based on flit time-stamp. After sorting the flits, the router allocates the output ports to the incoming flits based on their priority. The first step in port allocation, the router allocates as many uncontended productive ports as possible to the flits. The second step is if the flit has not been assigned any output port, the flit is assigned an output port if the port has not already been allocated by a higher priority flit and higher ranked flits are not deflected. if either conditions is fault, the flit is deflected. The latency of the critical path of the parallel port allocation is half that of the sequential port allocation.

Carpool reaches higher saturation throughput than BLESS. The average network latency is lower than BLESS. Forking the flits reduces the average latency and the network congestion despite producing high deflection rates. Without forking the network saturates at 0.12. The router is synthesized with Cadence Encounter at 35 nm standard library.

3.6 Hoplite

Hoplite designs a lightweight crossbar that suits the nature of the FPGA. This is the only one that targets FPGA platform. Hoplite [40] is a lightweight efficient buffer-less router. The router deals on flit basis. The assembly of flits is not addressed in [40]. The single flit carries the destination and the payload. BLESS occupies less area than buffered routers but it still occupies large area as it does not target the FPGA platform. Basically buffers and crossbar occupy the heavy resources in any network on chip router.

3.6.1 The crossbar

It is observed that changing the network topology or modifying it could simplify the crossbar greatly. This is actually what simplifies the permutation network in CHIPPER. Setting all the output ports valid even for the routers at the edges simplifies the permutation network in the 2D network from three stages to two stages only. Similarly, modifying the network topology here in Hoplite from mesh to torus simplifies the crossbar significantly. Using directional network such as torus reduces the complexity. The routing function in certain region could be changed in order to throttle the injection rate. This property takes advantage of the re-configurability.

Figure 3.9 shows how hoplite tends to decrease the complexity of the crossbar. Unidirectional torus reduces the crossbar complexity from 5x5 to 3x3. Hoplite uses dimension ordering where the input ports are statically ordered as shown in c. Hoplite removes the ejection port from the crossbar as illustrated in d. Hoplite removes the pipeline register used after the injection queue as shown in e. Finally, Hoplite uses the pipeline registers after the multiplexers as illustrated in g. The pipeline registers are placed after the switch in order to make the two multiplexers and the register fit in a single 6-input LUT in a 5-5 mode on Xilinx FPGA. Since, the router is single pipeline router; the dominant contributor to the critical path is the long wire delay from one router to another. Floor planning generator produces constraint file that floor plans the cluster.

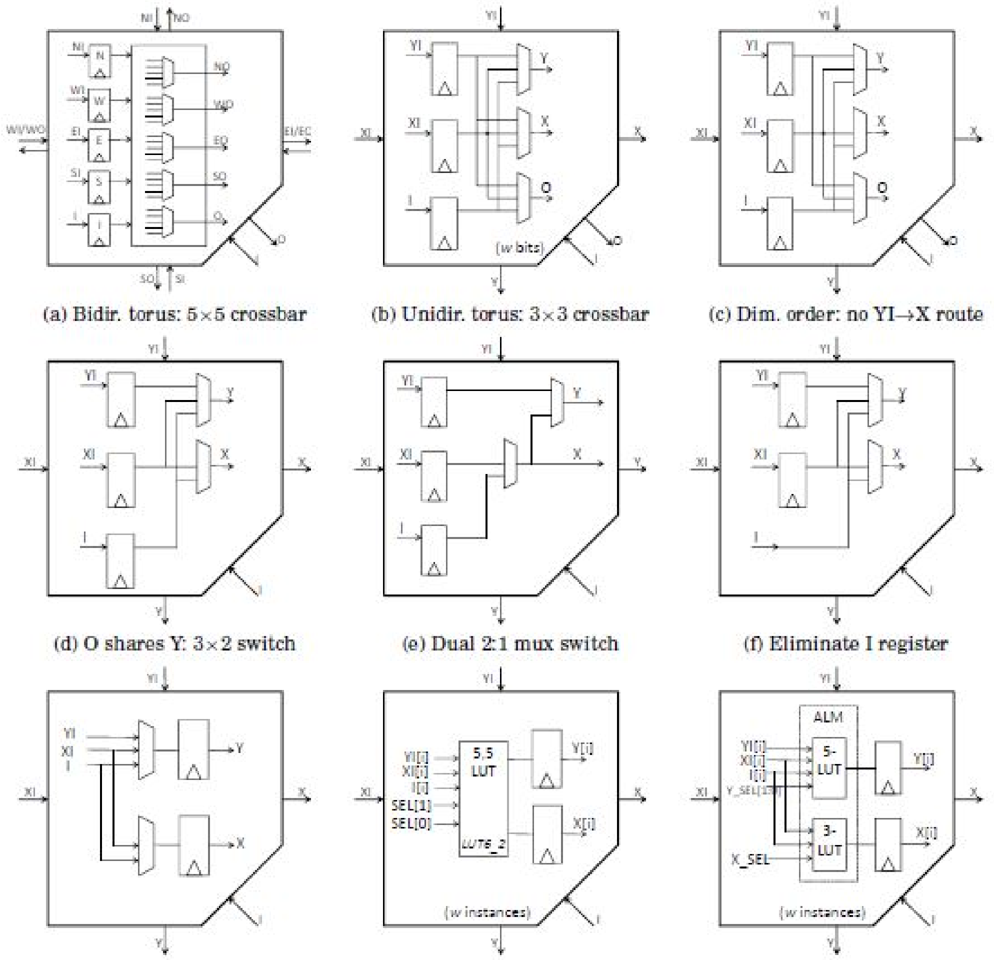


Fig. 3.9: Hoplite Crossbar [40]

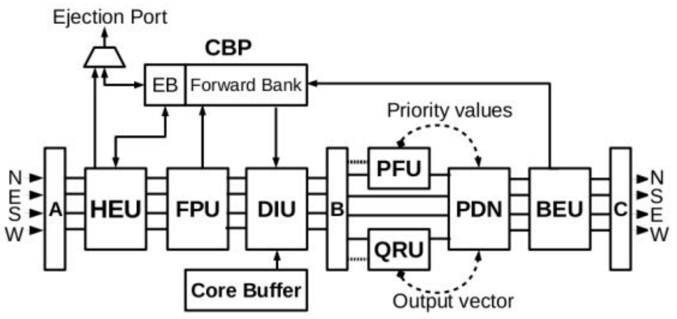


Fig. 3.10: Debar Architecture [25]

3.6.2 Result

Hoplite achieves lower saturation throughput than buffered routers. It has higher latency than buffered routers. However, it has higher operating frequency. Hoplite occupies less LUT resources for small networks. ”Deflection torus is only superior to the mesh for small PE sizes below 2K LUTs”.

3.7 Debar

Debar is a 2-stage buffer-less router. Debar [25] uses a central buffer pool to hold a fraction of the deflected flits. Debar proposes a new adaptive routing algorithm and a selective flit buffering based on flit marking. Resolving arbitration by random selection of flits at various stages of the router pipeline in MinBD and Chipper affects the network latency badly.

Debar shows a lower average latency than MinBD. Debar achieves less deflection rate as compared to MinBD for all synthetic traffic patterns. This is due to the priority scheme that prevents the deflection of flits once they are near to the destination.

Debar is synthesized by synopsys design compiler with 65nm library. Debar without the dual ejection unit property has the same router area and power dissipation as Minbd. Debar and MinBD can work on the same operating frequency.

To model two cycle deflection router Chipper for experimental analysis, Booksim simulator is used.

Router architecture:

A hybrid flit ejection mechanism that gives the effect of dual ejection with a single ejection port. **Hybrid ejection unit** identifies the flits intended to the local core. When there is a single ejection flit in the current cycle, the flit is removed from the internal flit channel and is forwarded to the ejection port. If the Ejection Bank in the central pool buffer is empty, HEU can handle at most two flit ejections at the same cycle.

Dual Injection unit (DIU):

Debar can inject from both the central pool buffer and the core buffer. Debar uses double inject unit DIU to inject flits.

Flit preemption unit (FPU):

When the channels are busy for a while (2 cycles), both the central pool buffer and the core buffer saturate. As the central pool buffer saturates, the deflection rate increases. To prevent

this saturation, the router uses flit preemption unit. When either the core buffer or the central buffer pool reaches a certain threshold. The router preempts one flit from the internal flit channel and places it in the central buffer pool. By this preemption, an empty slot is created in the internal flit channel so that the injection and re-injection from the respective buffer can be resumed, thereby avoiding starvation.
Priority fixer unit (PFU):
Debar prioritizes flits that are nearer to the destination. The flits are sorted into three levels.
0-flits whose destination are within 2 hops.
1-flits whose destination are between 2 and 4 hops.
2-flits whose destination are more than 4 hops away.
Quadrant routing unit (QRU):
Based on the destination address of the flit, an output vector is computed. It indicates the possible productive ports for the flit. If the source and destination are on the same row or colomn. The flit will have one productive output port. However, if they have different row and colomn therefore it will have two productive output ports.
Permutation deflection network:
It is the same unit used in Chipper and MinBD. However, Debar has two additional control units to it. The header enhancer circuit that adds the priority value and the output vector to the flit header, and the flit marking circuit that identifies the misrouted flits from others. It compares the allocated output port with the productive output port attached to the header. Flits marked with one indicate that they are assigned non-productive which take them away from destination.
Buffer ejection unit(BEU): selects at most one flit marked with one for storing into the forward bank of CBP. This is just like the side buffer used in MinBD in order to reduce the deflection rate. Once the flit leaves the router to the output ports, the marking bit is cleared.

The depth of the central buffer pool in the routers at the centre of the mesh network is 4, at the edge of the network is 3 and at the corners of the network is 2. The corner and edge routers carry less traffic than the centre routers. The simulation is conducted on 8x8 network.

3.8 SLIDER

Smart late injection deflection router uses side buffer for accommodating a fraction of deflected flits. The main contributions are smart late injection and selective flit preemption.

Smart Late Injection: Slider [19] was used to enhance the performance of Debar. First the channel wastage in Debar. A case in Debar when all the four input channels of a router are busy, the local inject unit can not inject. When a new flit is injected after permutation deflection network (where the flits are allocated), if it does not get its productive output port, it may be selected out of the router pipeline and buffered in the central buffer pool. Slider gets rid of this inner cycle in order to save the dynamic power. Slider uses the injection to the end of the router pipeline to utilize the idle output channels already existing in the router pipeline or created by flit preemption. The channel wastage is 18% for Debar, by using late injection in Slider the wastage drops to 6% only.
If this newly injected flit is selected after the permutation network, it will be buffered in the central buffer pool. The movement from one buffer to another leads to unnecessary power consumption without any forward progress for the flits. Sliding the injection logic at the end of the router pipeline prevents intra router movements completely. The late injection tackles live-

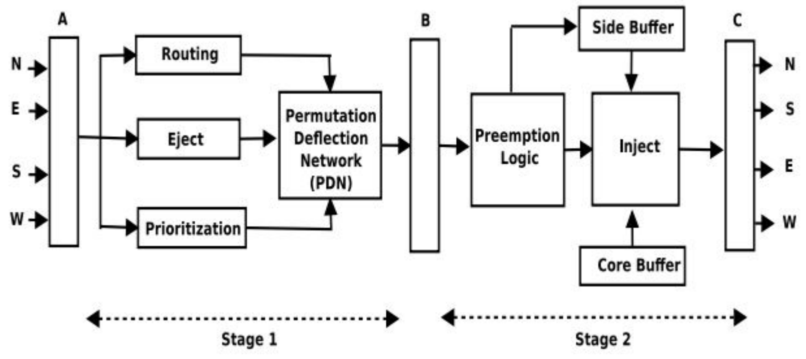


Fig. 3.11: Slider Architecture [19]

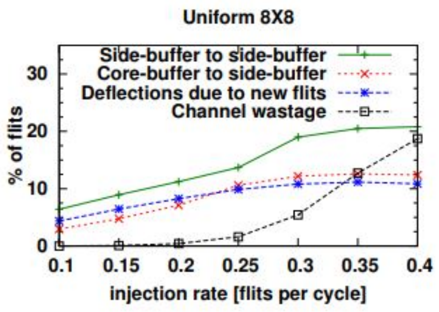


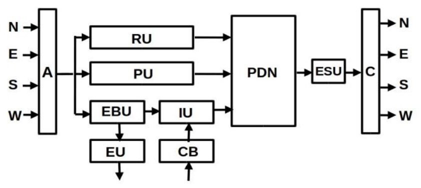
Fig. 3.12: Percentage of deflected flits [19]

lock problem that happens in these situations." When these high priority newly injected flits can have port conflicts with incoming older flits from the neighbors and the older flits may get deflected"[19]. From figure 3.12 it appears that 10% of the flits are deflected because of high priority new flits.

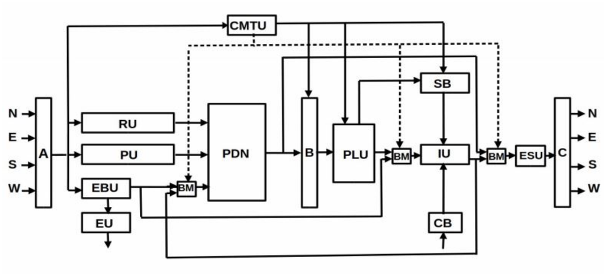
Selective Preemption:
While MinBD uses two eject units and Debar uses a dual ejection with a single ejection unit, Slider ejects at most one flit each cycle. Slider uses XY routing algorithm and a hop based priority scheme. Slider uses selective preemption of flits. Selective preemption is the process of preventing a flit from moving out through its assigned output port. This preemption is done to prevent a flit from moving out through a nonproductive port or to make space for a starving flit waiting in the router buffers either core buffer or side buffer. When the number of flits exceeds a certain threshold, Slider injects flits into the available channel irrespective of whether they have productive port or not. This is called non-restricted injection. On the other hand, restricted injection injects flits only if a productive channel is available. It may slowly increase the number of flits waiting in the respective buffers; but it results in proper utilization of the channel. The threshold is chosen to be 2 flits per buffer as it gives the furthest saturation point for the given circumstances.

Parallel Operations: These three operations, ejection, routing and prioritization. Three of them are independent, therefore,Slider makes them work in parallel.

Slider occupies less area than DEBAR and MinBD. Also, it consumes less static power consumption. It has higher saturation point than Debar and MinBD. Slider is capable of supporting higher injection rates and higher traffic. The deflections decrease with respect to Debar and MinBD. Therefore, the average latency decreases than both routers. Slider supports higher op-



(a) Single Pipeline



(b) Two Stage Pipeline

Fig. 3.13: Cascade Router Architecture[39]

erating frequency than Debar and MinBD. Also, Slider can deal efficiently with different traffic forms.

3.9 Cascade

The main idea is to design a router that dynamically configures itself based on the router congestion level. The router [39] switches from single-cycle buffer-less router to two cycles minimally buffered router and vice verse. Each router can switch independently. There is no switching overhead.

At low loads, a simple single cycle router is preferred over k cycle router. Because it can reduce latency, area and power consumption. At high loads, as port contention can be high, a simple single cycle router without any flit management provisions leads to heavy deflection rate, longer delay, and increased network activity, which in turn cause early saturation. The router uses two stages at high load. The side buffer is used only at high loads. At low loads side buffer is underused, so it is bypassed in Cascade design.

Slider and Debar are two-cycle pipeline routers. Thus, Cascade manages flits more efficiently adapting to the traffic, accommodates more load and extends the saturation point. The router reduces power consumption by 19% on average with area overhead 9%.The deflection rate of the Cascade router is high compared to other routers initially. As the injection rate increases further, the deflection rate reduces further and becomes stable, unlike networks of other routers where the deflection rate increases gradually at initial periods and exponential afterwards.

Router architecture:

The router takes the occupancy bit of each channel from the pipeline register, as an input and monitors the congestion level in that particular router. Every input port has an occupancy bit which is set when the channel is occupied, otherwise it is reset. The router decides whether it is congested or not by two threshold values. If the number of busy channels exceeds a threshold (CT) during certain window (CW), the congestion monitoring and triggering circuit (CMTU) enables the two stages. If it is less than a threshold (CFT), it is congested and in the single-cycle mode, the router (CMTU) enables the trigger to convert it into the two cycle mode. Under single cycle mode, pipeline register B is bypassed and power-gated. Routing logic(RU) in the router uses X-Y routing.The router computes the priority for each incoming flit in the prioritization unit (PU) based on hops to the destination.

Eject bit set unit (ESU): It sets an eject flag, for the outgoing flit. If the assigned port is the same as the productive port and it is one step away from destination.
Eject Bypass unit (EBU): It checks the eject flag of the incoming flit. If the eject flag is set. It bypasses input flits with ejection flag set to the eject unit (EU) and sets the occupancy bit of this channel; if more than one flit needs to be ejected, one is chosen randomly.

The router performs port allocation by permutation deflection network. The permutation network has 4 arbiters. The router uses a side buffer(SB) to store the preempted flits. Centre buffer(CB) is used to store the flits injected by the local node. Both, the side buffer and the centre buffer are non-FIFO queues. Thus, every flit in these buffers are given a chance to be injected. This is implemented by using a 2-bit flag for each flit in these buffers to represent the desired output port.

PLU is the preemption logic unit one among the deflected flits is buffered in the side buffer. BM is a bypass multiplexer. It receives the flits and directs them based on the select trigger from (CMTU) unit in order to facilitate switching between the two modes.

Under single cycle buffer the preemption logic is bypassed and power-gated. The preemption logic forcefully preempts a flit from the router pipeline to allow the starving flits in the core buffer to be injected. Injection unit in this router injects from both the side buffer and the center buffer. If more than one link is idle, both buffers can inject. In the center buffer, if the number of flits is less than half its capacity, the router injects flits only if they got their productive output port. If the number of flits is more than half its capacity, the router injects flits even in non-productive directions.

3.10 Q-BLESS

This router is based on BLESS router. It enhances the performance of the deflection based router. As buffer-less router occupies less area and consumes less power. Q-bless adopts source throttling to control congestion. Q-bless prioritizes the running applications on the network. Flits of high priority applications are never throttled. Source throttling means that the node is ordered to stop injection when the network is congested, and the other starving nodes are allowed to inject.

During each epoch, each node calculates the congestion by recording the throttling rate and counting the number of starvation cycles, if the node is starving. A node is considered starving if the time period at which the node has not inject any flits exceeds a certain threshold. Each node has its own threshold. A global controller gathers this information, and determines the throttling rates. The global controller selects the nodes that run low priority application to throttle. This picky throttling improves the performance at high load in buffer-less network. It has a higher saturation point than simple source throttling.

3.11 Opensource simulators

3.11.1 Noculator

”NOCulator is a network on chip simulator providing cycle accurate performance models for a wide variety of networks (mesh, torus, ring, hierarchical ring, flattened butterfly) and routers (buffered, bufferless, Adaptive Flow Control, MinBD, HiRD).” [21][28]. Noculator provides the RTL of buffer-less BLESS and Chipper and a buffered design. It also provides a simulator in C# for all the above topologies and routers.

3.11.2 Booksim

A flexible simulator with the following parameters topology, routing algorithm, flow control and allocation scheme[27].

3.12 Network metrics in Buffer-less Networks

In buffer-less networks the router can not inject unless there is free channel. Therefore, the latency does not increase with the increase in congestion, in case that the latency is calculated from the time the flit is injected not from the time it is generated. The proper metric to represent congestion at each node is the injection starvation. Injection starvation is the number of cycles the router could not inject in.

[22] mentions that the flits are not created equal. The throttling of applications have different impact on the overall network throughput. The author conducts an experiment of two applications. The author concludes that the control mechanism on application level should have instruction/flit IPF. As the throughput of flits depends on the cache miss rate of each application. However, this work has nothing to do with congestion control on application level.

Chapter 4: Buffer-less Modified CONNECT Router for 2D and 3D NoC

This chapter discusses Modified CONNECT. Modified CONNECT is the main contribution of this work.

4.1 Modified CONNECT

Modified CONNECT [47] is a lightweight and efficient buffer-less router. Modified CONNECT is a modified version of CONfigurable NETwork Creation Tool(CONNECT). Modified CONNECT saves 30% area compared to CONNECT while employing competitive performance. The router is compared against the available buffer-less routers BLESS and CHIPPER.
At each clock cycle, the router could receive new flits at each input port. At the end of each cycle, the flits should be delivered despite contentions. Modified CONNECT has the same pipeline architecture as that of CONNECT 4.1. However, at each input port the buffers are replaced with pipeline registers. Similarly, the output FIFO module is replaced with pipeline register at each input port. The input pipeline register senses whether there is a valid incoming flit. Input flit fetches the look-up table using destination to get its output port. The output port is appended to the flit. The flit is saved in the pipeline register and the appropriate output port is saved in its pipeline register. The flit header (the requested output port) and the input buffer occupancy are fed to the allocation logic to arbitrate for the appropriate output. It is worth mentioning that there is no credit handling or backpressure in buffer-less designs except at the local (injection) node. CONNECT can eject one flit only per cycle.

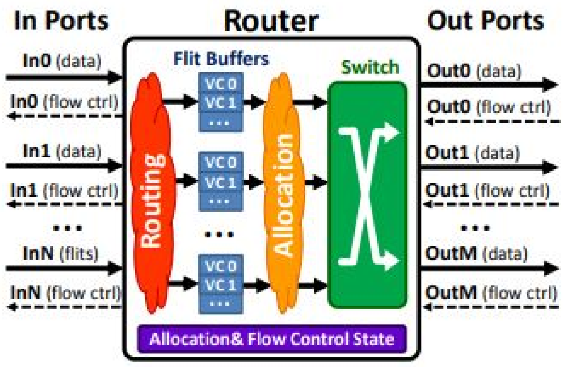


Fig. 4.1: CONNECT Router [45]

Allocation is divided into two sequential stages. Eject logic unit uses round robin to give priority to the input ports. Static input arbiter is removed as it is useless in this design specification(mesh network). Static output arbiter is used as it occupies less area utilization than round robin arbiter. Static output arbiter is used in order to maximize the number of flits directed to their productive direction. This simple fixed priority grants the highest priority flits their requested output ports. As each flit arbitrate to its desired output port. The flit of the higher priority is granted its desired output port. Each output port has a fixed sorting for flits of each input port; flits that are not assigned any output ports are forwarded to the next stage.

The allocation logic of Modified CONNECT Fig.4.3 improves the performance as it allows the maximum number of flits to be directed to its productive output port and it prevents unnecessary deflections. Reducing the deflection rate, reduces also the dynamic power dissipated as well.

At contention, flits of lower priority are not granted any output port, therefore they are directed to deflection logic. The available output ports are directed to the deflection logic as well. The available output ports propagate vertically as shown in Fig.4.2, until they meet the requests from the incoming input ports. The requests are kept on the diagonal to keep the delay of propagation delay symmetric. The deflected flit is granted an output when the request which travels horizontally meets an available output port. The deflection unit allocates free ports to the unassigned flits through an efficient parallel way.

The router injects flits whenever there is at least one free channel after the allocation of output ports to the incoming flits. The inject unit injects flit even it is not in its productive direction. The late injection of flits decreases the average latency and increases the average network throughput. It also prevents any inner loops and dynamic power dissipation.

CONNECT and Modified CONNECT use encoder logic as a crossbar. Encoder logic occupies less area than traditional switches. The flit payload does not propagate down the router pipeline. Only header arbitrates for the output ports. Finally, the flit traverses a multiplexer to the output port. Five multiplexers are used to allow the pipeline registers to receive new input flits; and to allow these flits reach the neighbor routers.

A comparison is conducted between distributed RAMs and registers in ordet to choose the appropriate implementation of the pipeline register. It is found that for a single pipeline register of wide data path (around 71 bit wide), registers are more efficient because distributed RAMs are consumed even if they are not totally occupied. However, it is found that for long buffers it is more efficient to use distributed RAMs as found in [34].

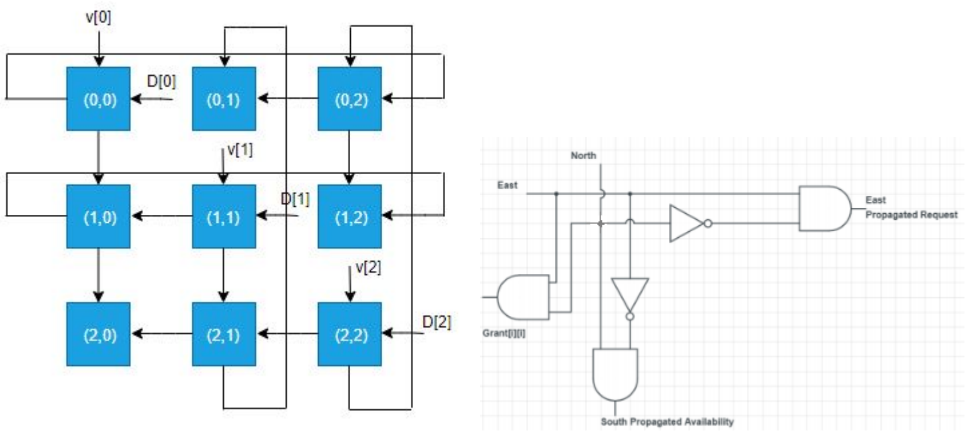
Modified CONNECT implements the pipeline registers as hardware registers. Modified CONNECT does not use distributed RAMs nor block RAMs in implementing the pipeline registers. Modified CONNECT uses the discrete registers in each slice. However, Modified CONNECT implements the injection node buffer as distributed RAM.

Modified CONNECT is a single pipeline router. FPGA operates at lower frequency than ASICs. Modified CONNECT overcomes this drawback of FPGA, by using wide datapaths. This shallow pipeline decreases the average network latency as it shortens the critical path of the router. The shallow pipeline also consumes less number of flip flops.

Modified CONNECT is evaluated against two metrics performance and hardware cost.

The allocation unit is implemented by another architecture. The second architecture is a permutation network 4.4. It is totally connected permutation network that uses fixed sorting in each arbiter. It is deadlock free. Each arbiter has two inputs. There are 6 arbiters in 3 stages. The architecture is based on the architecture of CHIPPER. However, that of CHIPPER is more area efficient. This architecture achieves exactly the same throughput and latency as the first allocation architecture. It occupies the same area.

Figure 4.5 is a walk through example to clarify the pipeline of Modified CONNECT.
At the first instant, the flit accesses the logic tables with the address of the destination (ADDR_1) and return back the output port number (D_OUT_1). At the same instant, Router0



(a) Deflection Unit **(b)** Single Element

Fig. 4.2: Deflection Unit

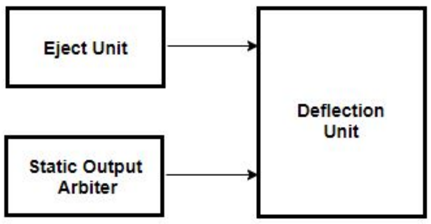


Fig. 4.3: Proposed Allocation Logic

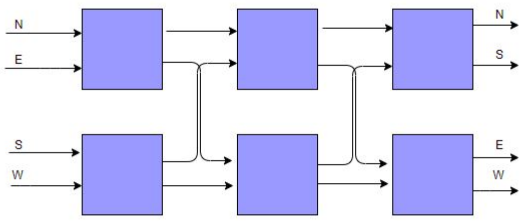


Fig. 4.4: Another Architecture to the Allocation Unit

receives a flit at input port 1 (in_ports_1_putRoutedFlit_flit_in).
(in_ports_1_putRoutedFlit_flit_in) is the input to the router and it is the concatenation of the (D_OUT_1) with the flit. The flit is saved at the (flitBuffers_1) pipeline register and the output port is saved at (outPortFIFOs_1) pipeline register. At the next clock cycle, the router concatenates the output of (flitBuffers_1) register with the output of (outPortFIFOs_1) register in (hasFlitsToSend_perIn_1$wget); and the flit arbitrates for its desired output port. The input for the allocation logic is (allocate_alloc_input)signal. In (allocate_alloc_input) the first five bits refer to the output port requests of the incoming flit from input port 0 and the second five bits refer to the requests of the incoming flit from input port 1 etc. In this example flit at input port 1 requests output port 4, therefore the second five bits in (allocate_alloc_input) are 10000. As the first bit is the request to output port 0, the second is the request to output port 1, the third is

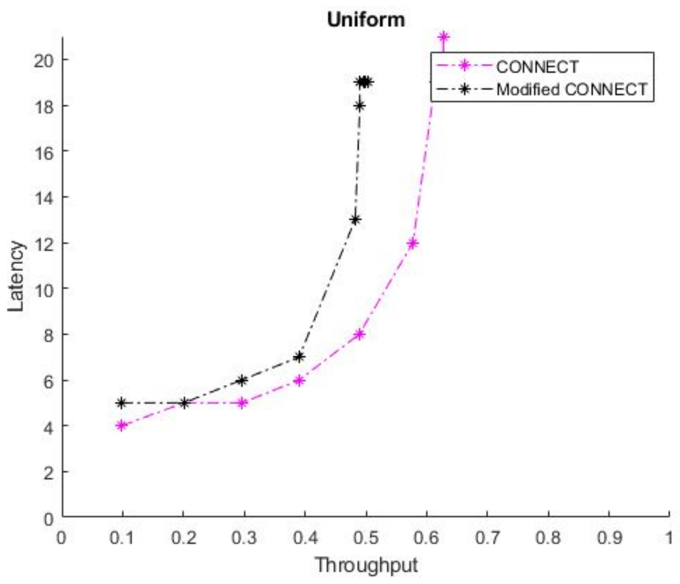


Fig. 4.6: Load Latency Curve

the request to output port 2 etc. There is no contention in this example therefore, the output of the allocation unit is the same as its input. CONNECT has five encoders. One for each input port. Each has five input ports. The output of input port 1 encoder is (outputencoder__d963). The highest significant bit is used to dequeue the input port buffer. The next three bits refer to the output port that the flit should be directed to, which is port 4 in this example . Output port 4 is (out_ports_4_getflit) signal.

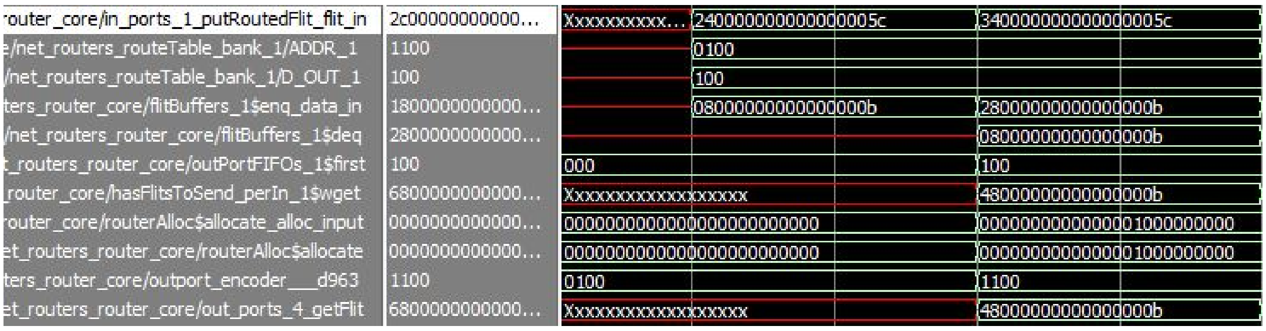
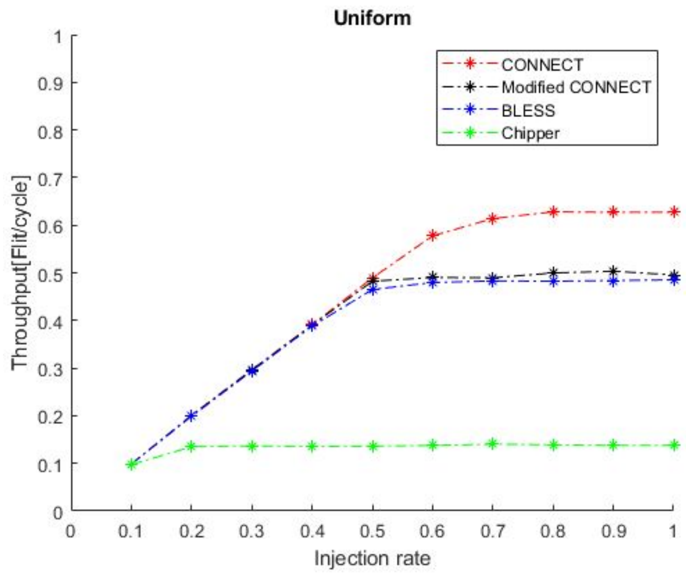

```
outer_core/in_ports_1_putRoutedFlit_flit_in   2c00000000000...   Xxxxxxxxxx...  240000000000000005c   340000000000000005c
/net_routers_routeTable_bank_1/ADDR_1          1100                              0100
/net_routers_routeTable_bank_1/D_OUT_1         100                               100
ters_router_core/flitBuffers_1$enq_data_in     1800000000000...                  080000000000000000b   280000000000000000b
/net_routers_router_core/flitBuffers_1$deq     2800000000000...                                        080000000000000000b
_routers_router_core/outPortFIFOs_1$first      100                000                                  100
router_core/hasFlitsToSend_perIn_1$wget        6800000000000...   Xxxxxxxxxxxxxxxxxx                   480000000000000000b
outer_core/routerAlloc$allocate_alloc_input    0000000000000...   0000000000000000000000000            0000000000000001000000000
t_routers_router_core/routerAlloc$allocate     0000000000000...   0000000000000000000000000            0000000000000001000000000
ters_router_core/outport_encoder___d963        1100               0100                                 1100
t_routers_router_core/out_ports_4_getFlit      6800000000000...   Xxxxxxxxxxxxxxxxxx                   480000000000000000b
```

Fig. 4.5: Modified CONNECT waveform

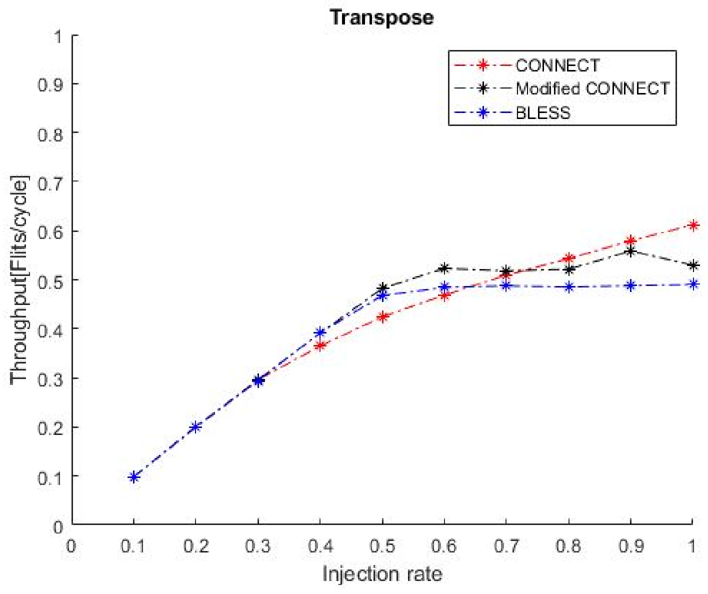
4.1.1 Performance Analysis

Basically, buffer-less routers have high performance close to conventional routers at low traffic; however, the performance degrades at high traffic loads. Modified CONNECT has competitive throughput as shown in Fig.4.6.

Modified CONNECT is evaluated against the available buffer-less designs BLESS and Chipper. The designs are evaluated using uniform, transpose and inverse traffic patterns. The RTL of BLESS and CHIPPER routers is available at [21]. The simulation is conducted on 4x4 mesh network with the same testbench used in simulating CONNECT and Modified CONNECT. The simulation is carried for 3000 cycle. Modified CONNECT has lower throughput than CONNECT. The same throughput as BLESS and higher than CHIPPER as illustrated at Fig.4.7. However, Modified CONNECT has higher maximum operating frequency than BLESS. Mod-



(a) Uniform Traffic Pattern



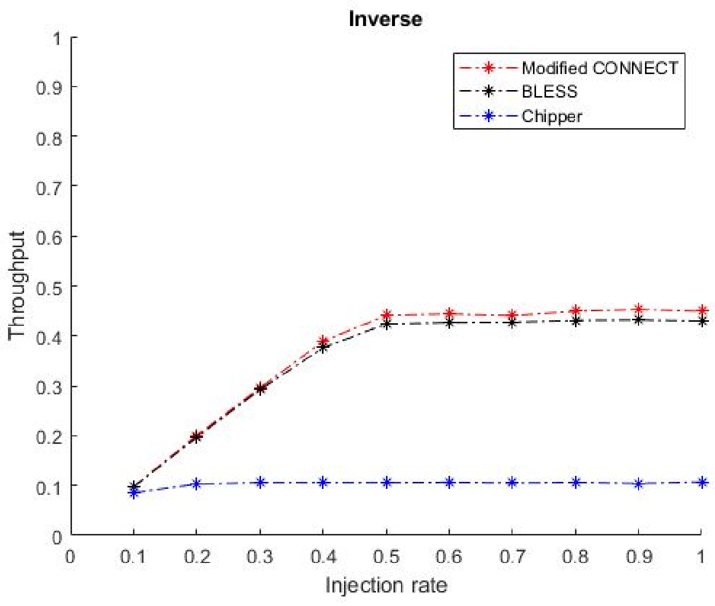
(b) Transpose Traffic Pattern

ified CONNECT has lower average latency than BLESS as illustrated at Fig.4.8. Modified CONNECT has the lowest deflection rate as illustrated in Fig.4.9.

4.1.2 Hardware Analysis

The four designs are synthesized using Virtex 7 evaluation kit(xc7vx485tffg1761-2). The area utilization of the whole 4x4 network is illustrated in Table.4.1. The percentage of the reduction in the total area is calculated using the resource utilization table in [34]. The resource utilization table in [34] is calculated for Virtex 5. This work assumes that the register area is approximately 3.6 times the LUT area for Virtex 7.

Modified CONNECT has 30% area less than CONNECT. Modified CONNECT reduces area compared to BLESS by 24% and reduces the area compared to CHIPPER by 18%. The reduction in the area of Modified CONNECT is due to the routing tables algorithm and the shallow pipeline. Dynamic power is calculated at maximum injection rate. Though the dynamic



(c) Inverse Traffic Pattern

Fig. 4.7: Traffic Evaluations for Network-Level Throughput

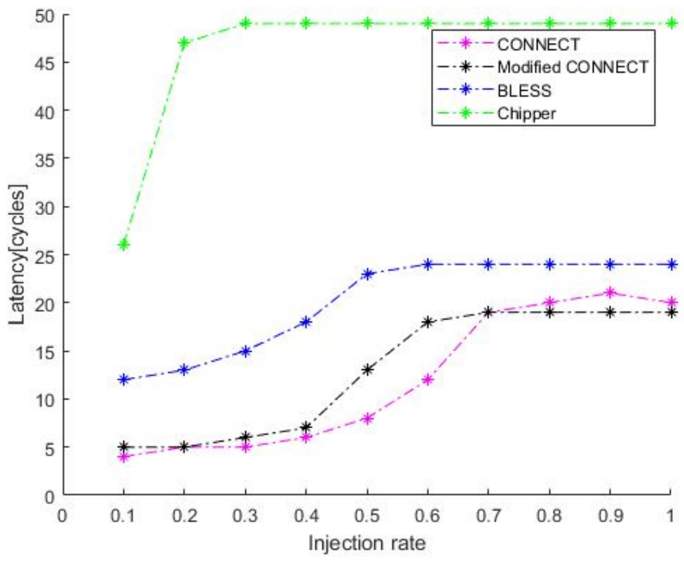


Fig. 4.8: Average Network Latency vs. Injection rate

Table 4.1: Area and Power Analysis

Router	LUT	register	Dynamic Power
CONNECT	14211	2624	0.003 (W)
Modified-CONNECT	8850	4144	0.035 (W)
BLESS	24814	33584	0.34 (W)
CHIPPER	20852	26784	0.075 (W)

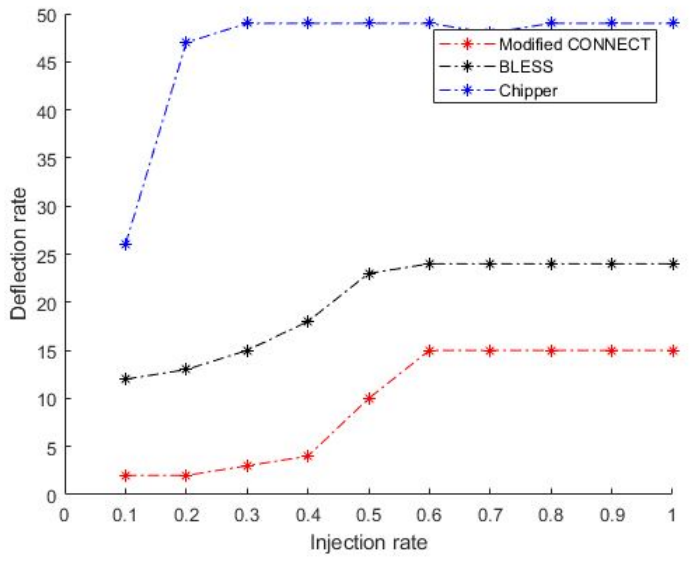


Fig. 4.9: Deflection rate for Buffer-less Designs

power in Modified CONNECT is higher than that of CONNECT for 4x4 mesh as shown in table4.1, Modified CONNECT scales better. In 12x12 mesh the dynamic power of CONNECT is 3.375 and that of Modified CONNECT is 0.769. Modified CONNECT keeps the same area savings at high dimensions. This is to clarify why Modified CONNECT has greater number of registers than CONNECT. The registers in CONNECT are used in the credit handling flow at each input port, they are used to control the FIFOs of both (flits and output ports) at each input port. Though the FIFOs are implemented as distributed RAMS, there is still a need for empty, full etc registers. All the previous registers are not used in Modified CONNECT, however Modified CONNECT saves the incoming flits from the four input ports in registers not in distributed RAMs. The number of registers in Modified CONNECT (in a single router) is approximately equal to the (number of ports -1) times the width of flit. The LUT area is approximately 3.6 times the register area, therefore the overall area occupied by Modified CONNECT is lower than than of CONNECT.

4.2 Introduction to Three Dimensional Networks

The speed of ASIC is higher than FPGA. ASIC occupies less area than FPGA. This is because standard cells have less area than look up tables. Because FPGA often needs to chain a large number of LUTs with long interconnects to emulate a logic block[45]. Also, the switches in FPGA consume more area and have larger capacitance than wires in ASIC[43]. The speed, area and power consumption of FPGA need to be improved. Three dimensional networks can improve the speed by 31% to 56% [43].

4.2.1 Performance and Dynamic Power

Three dimension network enhances the capabilities of stacking a larger number of processors. It increases the density of interconnects and IPs in the system. Two dimensional network is restricted by floor planning. Three dimension increases the capabilities to embed more IPs. 3D allows the possibility of integrating different technologies in one product. Another advantage of 3D is to provide substrate isolation between different blocks. Three dimensional network decreases the number of stages between the nodes, so the average throughput increases. The longest critical path shortens. It achieves higher performance and parallelism. Also less dynamic power. The consumed power per a single packet transmission is lower. 3D offers low cost scaling. The drawbacks of 3D are mainly the area overhead and the fabrication challenges in bonding the dies. Another restriction is the heat transfer. The temperature increases specially to the dies at the middle of the 3D structure.

The throughput of the 3D network is higher than that of 2D networks for these reasons [33]. First, for the same number of routers the number of links in 3 dimensional network is greater than that in 2D network. The number of links in 4x4x4 is 144. The number of links in 8x8 network is 112. Increasing the number of links increases the parallelism.

$$Links_{2D\ mesh} = N_x * (N_y - 1) \ + \ N_y * (N_x - 1) \tag{4.1}$$

where N_i is the number of routers in i dimension.

$$Links_{3D\ mesh} = N_x N_y * (N_z - 1) \ + \ N_y N_z * (N_x - 1) \ + \ N_x N_z * (N_y - 1) \tag{4.2}$$

The number of hops in 3D network is basically less than that in 2D network. The number of near neighbors in 3D network is greater. Therefore, in general the average number of hops in the path of a packet from source to destination decreases in 3D networks. This means that the average latency decreases in 3D network and the throughput increases. The latency improves in 3D NoC up to 33% compared to 2D NoCs.[46] The path from source to destination becomes shorter. The dynamic power consumed through the path from the source to destination decreases. It takes less power to transmit a single flit in a 3D network. However, the total power consumed in 3D network is higher because 3D network transmits more flits.

Finally, each router in the 3D networks has 7 ports. Routers in 2D networks has 5 ports. The number of flits that cross the crossbar is greater in 3D network. The greater number of ports in each router enhances the throughput.

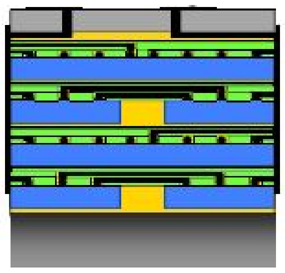
The delay of the interconnects is an important factor in the performance of the FPGA. The interconnect delay is more significant in the new technology. Since the transistor size scales down and the wire length remains as it is. The delay of the wires equals RC and the length of the wire remains as it is and the transistor delay decreases as the transistor scales down. However, the average wire length decreases in 3D network. The resistance of the wire with repeaters decreases. The capacitance of the wire decreases. The delay equals RC. Therefore, the delay of the wire with the repeaters decreases. The reduction in the wire delay contributes in the improvement of the performance. The dynamic power [26] also decreases as the wire length decreases, because the capacitance decreases as shown in eq.(4.3). Also the power supply decreases as a result of the scaling down of transistors in new technology generations.

$$P = \alpha . C_L . f . V_{dd}^2 \tag{4.3}$$

The zero load latency depends on the latency of the router and that of the channel. The zero load latency characterizes the performance of the 3D topology. If the channel delay is the dominant, it is better to decrease the channel length. If the delay of the router is dominant, it is better to decrease the number of hops that the flit traverses through its journey from source to destination. 3D routing reduces both the average number of hops and the wire length. Though the router delay increases as the number of ports increases, the increase in the router delay is small. The router delay increases logarithmically with the increase in the number of ports. The router delay is independent of the network size[46]. The 3D network enhances the performance and consumes less dynamic power per flit. It occupies more resources. 3D network that is discussed here is where the nodes are scattered in the three dimensions and the interconnects span the three dimensions.

4.2.2 Physical design

The manufacturing of 3D connections is illustrated in [32]. FPGAs are stacked on die or wafer scale. In Die Stacking the 3D connections are done through micro bumps. Wafer stacking the 3D connections are done using through silicon vias (TSV). The die stacking takes place by 3D package or by face to face micro-bumps. Micro bumps place gold bumps on the surface of each die. Micro bumps bring the signals to the edge of the die. The edge of the die that connects the tiers is metal. The 3D package offers high density connections. The bumps connect the signal from the edge to the IP. Face to face micro bumps shorten the wires and decrease the parasitic capacitance. Actually, what limits the number of tiers that could be used in 3D package is heat. The assembly process does not limit the number of tiers. The system is limited by the heat. Heat could degrade the system performance. High power designs need a cooling system with small area. The designs that benefit from the wire reduction are actually the hottest designs. This technology is limited to 2 tiers [32].


(a) Die stacking by Micro Bumps

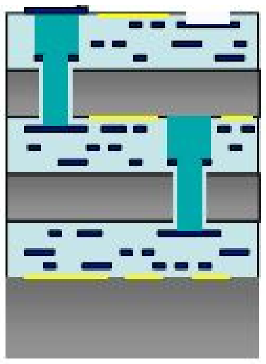
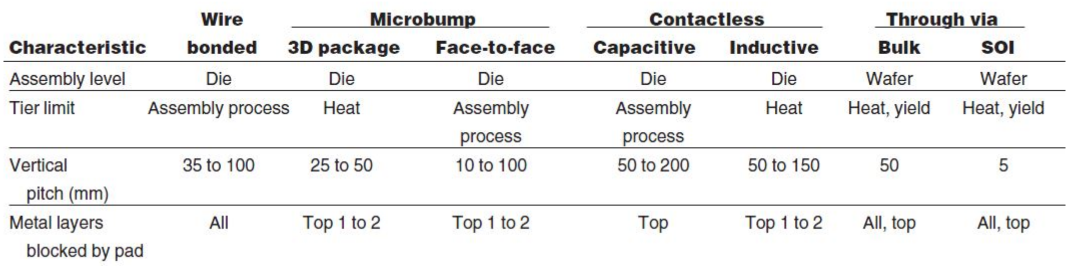

(b) Wafer Bonding by TSV

Fig. 4.10: 3D Manufacturing [32]

Wafer stacking uses the through silicon vias (TSV). Wafer stacking stacks the wafers together then cuts the wafers into dies. Wafer stacking has worse yield. The first wafer is placed, the second is placed face to face. The third is placed face to back and so on. Holes are created from the upper wafer to the lower. The holes are filled with tungsten for connectivity. The through silicon via enhances the connectivity. However, the increase in the number of tiers is restricted by the yield. As the wafers are assembled then cut, the vias decrease the wire length and so decrease the parasitic resistance and capacitance created by the wire. Therefore, the total average delay decreases. The decrease in dynamic power is more achievable than the increase in the speed. This is because of the parasitic capacitance and resistance of the via affects the speed. The mobility of electrons is affected by heat. Actually doubling the heat could produce 30% degradation in the performance. Fortunately, the reduction in the wire length makes up the heat effect on the mobility. Therefore, basically what limits the system is the yield. If the yield of the system is low, the cost of the system will be high. Beyond a threshold the yield decreases exponentially with the increase in the number of TSVs. The wafers are bound together before examining them. Therefore, a good die could be bonded with bad one. Though, vias have their parasitic capacitance and parasitic resistance, the parasitic capacitance is low. It does not affect both the delay and the power; however, it adds area overhead.



	Wire	Microbump		Contactless		Through via	
Characteristic	bonded	3D package	Face-to-face	Capacitive	Inductive	Bulk	SOI
Assembly level	Die	Die	Die	Die	Die	Wafer	Wafer
Tier limit	Assembly process	Heat	Assembly process	Assembly process	Heat	Heat, yield	Heat, yield
Vertical pitch (mm)	35 to 100	25 to 50	10 to 100	50 to 200	50 to 150	50	5
Metal layers blocked by pad	All	Top 1 to 2	Top 1 to 2	Top	Top 1 to 2	All, top	All, top

Fig. 4.11: Comparison of 3D connections [32]

3D adds area overhead because inter-layer vias need dedicated area as shown in Fig.4.12. Three dimension network uses either grid based topology or Fat tree topology. Mesh topology is the common used topology. It is implemented by Through Silicon Via (TSV). Tree topology layout divides the planar network into parts and connect them vertically as well.

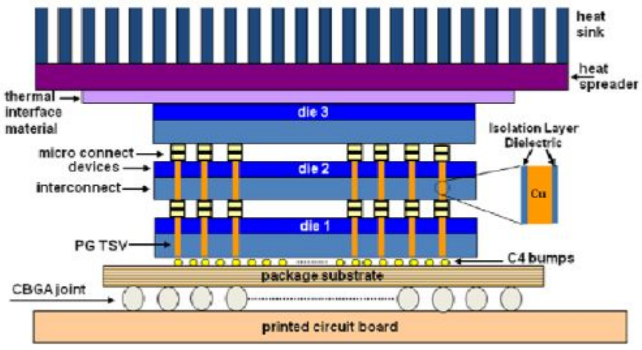
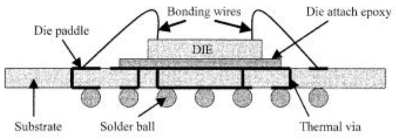


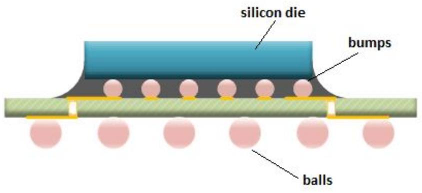
Fig. 4.12: Area occupied by TSV [42]

Overview on Packaging used in 2D IC

This is an overview on packaging of 2D ICs. First, wire bond ball grid array (BGA) Fig.4.13a is adopted. The wires are typically 15-35 μm. In order to increase the speed and to reduce wire inductance, flip chip connectivity is used instead of wire bonding. FlipChip offers high pin count, high signal density, better power dissipation, low signal inductance and good connectivity. Flip chip is not a package, it describes the method of electrically connecting the die to the package carrier. Flip Chip is used in order to increase the speed. It offers better power dissipation, low signal inductance and good connectivity. Flip chip uses conductive bumps, the bumped die is flipped over so that the bumps face the package carrier as shown in Fig.4.13b. After the die is soldered, underfill is used between the die and the substrate. The under fill is used to reduce the stress on the soldered joints. Flip chip package is an extension to flip chip where the package may contain multiple passive components. Silicon devices are bumped and embedded into 2 or 4 layer substrate. Flip chip family contains bumps with different shapes and sizes and different materials. Chip scale package is 1.2 times the size of the die inside. Chip scale package may



(a) 2D wirebond packaging[48]



(b) Flip Chip Connectivity[9]

Fig. 4.13: 2D Packaging

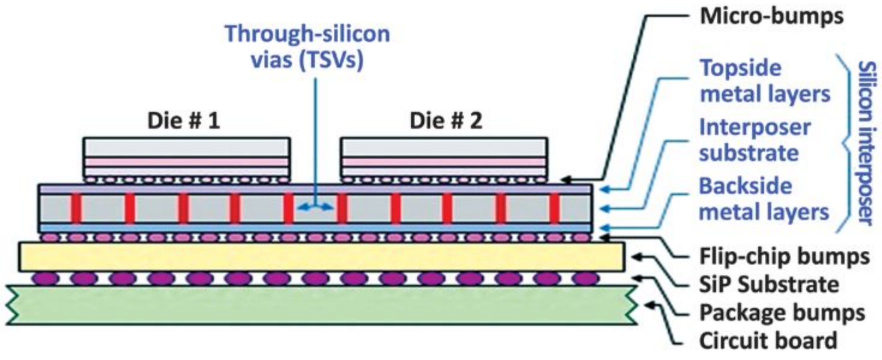


Fig. 4.14: 2.5 Packaging[5]

use wire bonding, flit chip or wafer bonding. Die stacked chip scale package is the process of mounting multiple chips on the top of each other. The chips used to be of different sizes and wire bonding was used to connect the chips. Package on package packaging is where individual packages are stacked on the top of each other. System in Package packaging is where multiple bare dice are mounted on a top of the same substrate to connect them all.

The packaging is upgraded to 2.5 packaging. It allows the usage of chips with different technologies. As chips are not connected directly to the substrate, there is an interposer between the chips and the substrate as shown in Fig.4.14. It decreases the wire length between multiple dies.

4.3 Three-dimensional Modified CONNECT

Three-dimensional network might be a reasonable solution to the performance limit. The performance degrades as the network diameter increases. The comparison between 8x8 Modified CONNECT and 4x4x4 3D Modified CONNECT shown in Fig.4.15 shows that the area overhead is small compared to the performance gain in 3D network. This metric (throughput/ area. power. latency) of 4x4x4 Modified CONNECT is higher than that of 8x8 Modified CONNECT by a decimal point.

4.3.1 Routing algorithm

Modified CONNECT is extended to be used in three-dimensional network. The router has seven ports with a single allocator and a switch. The router picks the z direction first. The router at each node decides whether the destination of the flit exits on the same layer or not. Then, it

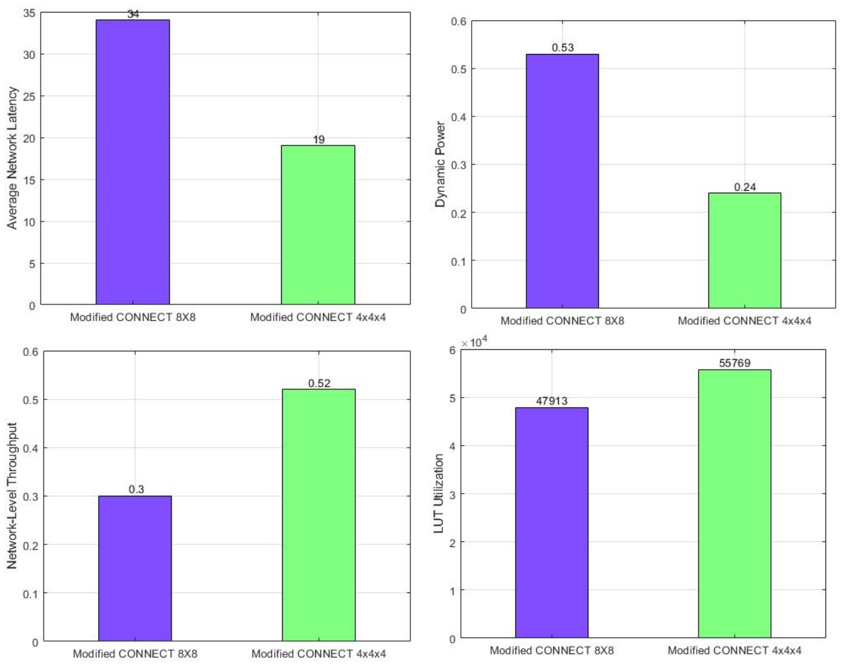


Fig. 4.15: Modified CONNECT in 2D and 3D NoC

prioritizes x direction over y direction if both are valid. Homogeneous 3D network acts as vertically connected layers.

4.3.2 Router Architecture

3D Modified CONNECT has 7 input/output ports. Modified CONNECT has 5 input/output ports. The input ports are East, West, North, South and node. 3D Modified CONNECT has two additional input/output ports which are up and down. This is the main difference between Modified CONNECT and 3D Modified CONNECT. The router uses routing tables that prioritize x direction over y direction if both are valid. Modified CONNECT prioritizes the x direction over y direction as well. The routing algorithm is implemented in both by routing tables not by routing function. Each input port has its distributed RAM. The distributed RAM is the routing table of this port; therefore, the number of routing tables per router is seven instead of five. The size of every single RAM increases as it has entries equal to the number of routers in the whole network. There is a single pipeline register for each input port. The allocator logic is divided into two blocks just like Modified CONNECT. The input to the allocator logic is 49 bit, seven input ports and each one of them has 7 bits to represent the output port that it requests. The first block is the static output arbiter in which each output port chooses an input port out of the input ports requests. This is implemented as 7 multiplexers each has 7 inputs instead of 5 multiplexers each has 5 inputs. The size of this block is insignificant, it is approximately 10 LUTs. The second block is the deflection unit. The deflection logic is waveform allocator. The deflection unit is 6x6 matrix of small arbiters. In Modified CONNECT the deflection unit is 4x4 matrix of small arbiters. The deflection unit works with the same logic used in Modified CONNECT. Finally, Modified CONNECT uses encoder to trigger each input port to receive new data. The encoder also enables a multiplexer which directs the flit to its allocated output port. Therefore, 3D Modified CONNECT has 7 encoders instead of 5 and 7 multiplexers instead of 5.
The flit width of 3D Modified CONNECT is greater than that of Modified CONNECT as the number of address bits in 3D Modified CONNECT most probably is greater than that of Modi-

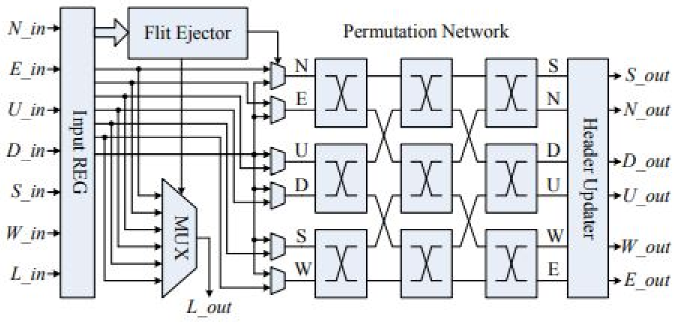


Fig. 4.17: 3D Buffer-less PERM [16]

fied CONNECT.

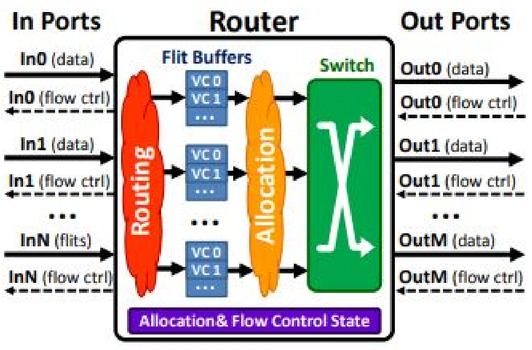


Fig. 4.16: CONNECT Router [45]

4.3.3 Discussion

It is clear from the previous discussion on CONNECT architecture that as the number of ports increases, the size of routing logic tables increases. Also, the number of logic tables per each router increases; because the number of routers in the network increases. The size of the allocator increases as well. The deflection matrix arbiter becomes 6x6 instead of 4x4 in 2D Modified CONNECT. The static output arbiter also increases. The size of the static arbiter is insignificant. It is approximately 10 LUTs. The number of encoders in the switch module increases. 3D Modified CONNECT offers flexible number of 3D routers. Any router could be 3D router or 2D router.

Three-dimensional Modified CONNECT is an efficient three-dimensional router. It outperforms the available 3D buffer-less routers. Below Modified CONNECT is evaluated against the available three-dimensional routers 3DAPBLESS [49], 3D-PERM [16] and 3D-BLESS. 3D-BLESS is the three-dimensional version of BLESS. 3D-BLESS operates at less than 100MHz.

3D-PERM [16] is a single cycle buffer-less router. Three-dimensional PERM adheres to the architecture of 3D-CHIPPER. The key difference between 3D CHIPPER and 3D-PERM is the way each addresses live-lock problem. 3D CHIPPER uses the golden flit rule, where pseudo random flit is chosen to be prioritized for a long enough time to be delivered to its destination. 3D-PERM sorts the incoming flits by the age priority. The oldest flit chooses its requested output port. 3D-PERM guarantees the delivery of the oldest flit only, as the router partially sorts the flits. The router uses the permutation network in sorting and arbitration. The permutation network has three stages of small units. Each unit has two inputs. The unit sorts the two incoming flits. The older chooses its requested output direction and therefore the unit either swaps the inputs or passes it. The permutation network of 3D-PERM is the same as that of 3D-CHIPPER. It is not fully connected; however, it deflects less number of flits. Because ”golden flit” sorting ensures that a single flit is directed to its productive output port. On the other side age priority guarantees that more than one flit is directed to its output port, the actual number of flits that this permutation network can direct it to its productive output port depends on the tree of the permutation network. The multiplexer shown in Fig.4.17 is the injection logic. Flits are injected whenever there is a vacant link. Flit traverses the permutation logic. It traverses the router pipeline. Comparing 3D PERM to 3D BLESS, 3D-PERM has a shorter critical path thanks to the permutation network. The permutation network solves the problem of the low operating frequency of BLESS.

Though the permutation network improves the critical path, the arbitration path still dominates the critical path. According to [49] the permutation network represents a 71% delay out of the delay of the critical path. The router critical path is ineffeciently long for most cases, as permutation networks are effiecient at low injection rate. At low injection the input of the permutation network is at mostly one flit. Therefore, 3DAPBLESS [49] replaces the large comparator inside each unit in the permutation unit with a smaller one. 3DAPBLESS compares only the five most significant bits in the two incoming flits. 3DAPBLESS_Lite is the second version of 3DAPBLESS. It compares only the most significant bit in the age field of the incoming flits. Both 3DAPBLESS and 3DAPBLESS_Lite outperform 3D-PERM, and they occupy less area.

4.3.4 Simulations and Results

Modified CONNECT, 3DAPBLESS, 3D BLESS and 3D-PERM are synthesized on Virtex 7 evaluation kit(xc7vx485tffg1761-2). Modified CONNECT, 3DAPBLESS and 3D PERM operate at 100 MHz. 3D BLESS operates at 10 MHz. 3D-PERM and 3DAPBLESS are the only two available 3D buffer-less routers. To evaluate 3D Modified CONNECT, it was essential to extend the design of both routers BLESS and CHIPPER to be used in 3D network. The RTL of 3D-PERM is not open source, therefore, its RTL is built by using the permutation network of 3D CHIPPER and modifying it. Similarly, the RTL of the pipeline of 3DAPBLESS is built. Modified CONNECT throughput is 27% higher than 3DAPBLESS throughput and 42% higher than 3D PERM throughput as illustrated in Table4.2.

The area occupied by each router is illustrated in Table 4.3. The area resources occupied by each router is calculated at its maximum operating frequency. These designs are implemented on Virtex 7. Virtex 7 is a 2.5 technology. It supports TSVs but not in the third dimension. The calculations might not be precise. However, the tool performs design route checks and total parasitic extraction after routing. Therefore, the delay and power calculations have taken the parasitics in their considerations. It is clear that though 3DAPBLESS can reach to a slightly higher frequency, Modified CONNECT achieves competitive frequency and it saves area resources significantly.

When Modified CONNECT operates at 125 MHz and 3DAPBLESS operates at 142 MHz, Modified CONNECT achieves 65.5 MHz and 3DAPBLESS achieves 67 MHz. In this case, Modified CONNECT occupies 54% area less than 3DAPBLESS. 3D BLESS has the lowest latency in ns and the lowest throughput. The estimated resources benefit obtained by 3D Modified CONNECT is calculated using area resources table in [34].

Table 4.2: Performance Analysis

Router	Maximum throughput (Flit/cycle)	Latency (cycle)	Maximum frequency (MHz)
3D Modified CONNECT	0.524535	21	125
3DAPBLESS	0.471854	23	142
3D-PERM	0.479521	23	100
3D-BLESS	0.497375	16	less than 100 MHz

Table 4.3: Area Utilization

Router	LUT	Register
3D Modified CONNECT	51783	17168
3DAPBLESS	107728	57936
3D PERM	115837	64562
3D BLESS	121017	129632

4.4 Flit Structure and testbench

CONNECT generator uses the flit with the below structure 4.18. The structure of the flit of BLESS is shown in Fig.4.19. The testbench consists of a packet generator that generates the packet according to the packet injection rate chosen by the user. The router sends credit to the credit handling that counts the empty slots in the injection buffer. If the number of empty slots is greater than 3 slots the credit handling module sends to the packet generator to send a new flit. The packet generator stops packet generation when the local injection queue is nearly full. The state of the injection queue whether it is full or empty is the starvation index. The starvation index is local, it does not represent the network state. The valid bit, tail bit and destination fields are filled by the packet generator unit. There are packet generator and credit handling modules for each router in the network. This testbench was designed in [31].

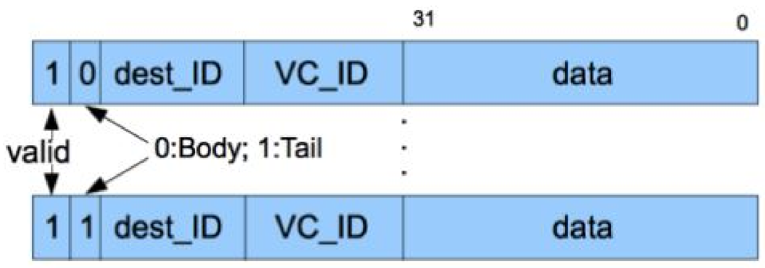


Fig. 4.18: Flit Structure of CONNECT

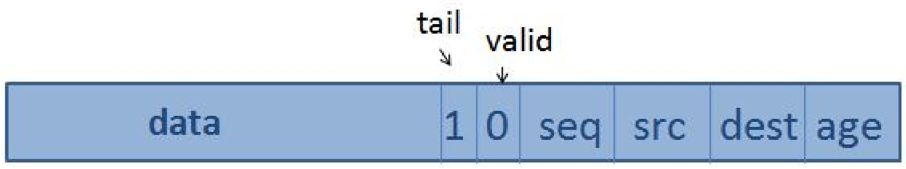


Fig. 4.19: Flit Structure of BLESS

In order to use the testbench with BLESS and CHIPPER. It is essential to use positive edge counter as both BLESS and CHIPPER are three stage routers. The RTL of BLESS is attached in the appendix. The backpressure signal is “port4_ ready”.

Chapter 5: CAD Tool for the 3D NoC Parameters Optimization

5.1 Optimum Size

In this chapter simulations are conducted in order to find the optimum size of a square mesh, the optimum number of layers, the optimum number of three-dimensional routers and their locations. The target is to generalize the optimum number of three-dimensional routers per mesh for three-dimensional buffer-less networks. The five buffer-less designs and CONNECT are evaluated against a chosen metric (figure of merit).

$$Figure\ of\ Merit = \frac{Throughput}{Area * Power * Latency} \tag{5.1}$$

The operating frequency of 3D BLESS is 10 MHz and the operating frequency of the all the other routers is 100 MHz. In the figure of merit (FOM), the throughput and the latency are in terms of cycles. The power term is the dynamic power. The static power is 0.24 (W) for all the designs.

5.1.1 Optimum Size of Square 2D Mesh

The FOM decreases as the size of the square mesh increases as shown in Fig.5.1. Because the area increases, the number of flits per network increases and therefore the dynamic power increases, the diameter of the tier increases and therefore the latency increases, and finally the throughput decreases as the core of the each single tier becomes more congested. The FOM saturates nearly at mesh size 4x4.

5.1.2 Optimum Number of Tiers

The single tier is 4x4 square mesh. Each and every router in the mesh is a 3D router. It is observed that the FOM decreases as the number of layers increases. Because the area occupied by network increases, the dynamic power increases, latency of flits increases and the throughput decreases in all the designs except in BLESS, it saturates. The FOM saturates after 4 layers.

It is noticed that placing the new IP cores vertically by increasing the number of tiers is better than placing the new IP cores horizontally by using a larger mesh network. It is noticed that the increase in mesh diameter, degrades the performance. The slope of Fig.5.2 is less than the slope of Fig.5.1. It is worth mentioning that the scale of Fig.5.2 is larger than that of Fig.5.1, because the number of routers in the network in Fig.5.2 is greater than that in Fig.5.1. For the same number of routers placing the routers vertically gives smaller decrease in the figure of merit. The 3-dimensional network could be a solution to the performance limit.

5.1.3 Optimum Number of 3D Routers

The target is to generalize the optimum number of 3D routers in a buffer-less 3D network. However, it is noticed from Fig.5.3 that, there is correlation between the results of CONNECT and

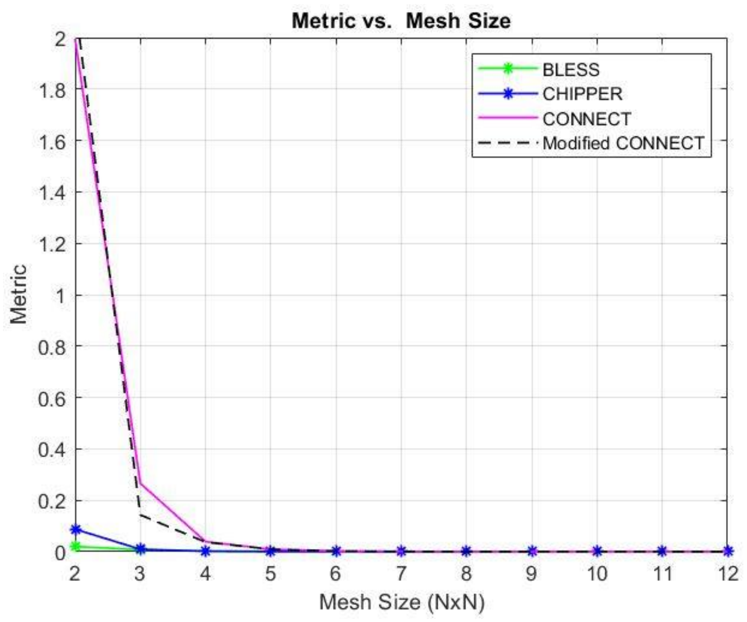


Fig. 5.1: Optimum Size for Square 2D mesh

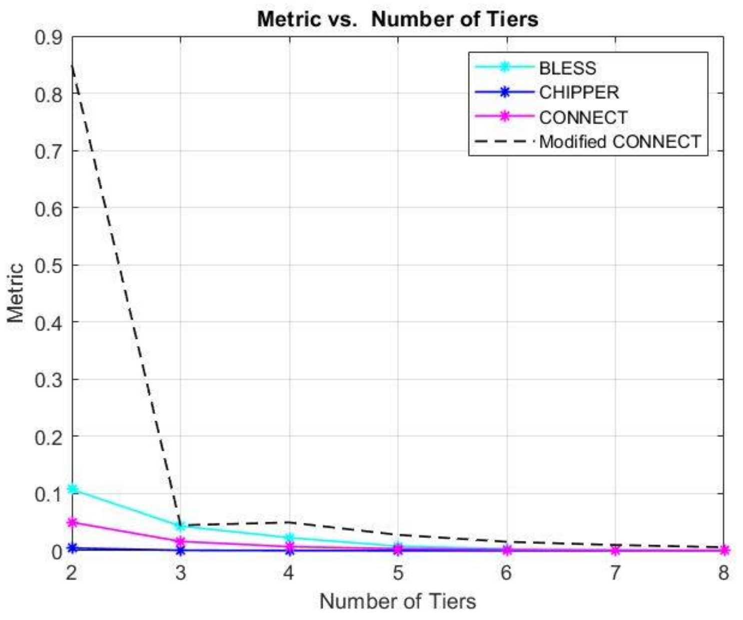


Fig. 5.2: Optimum Number of Tiers

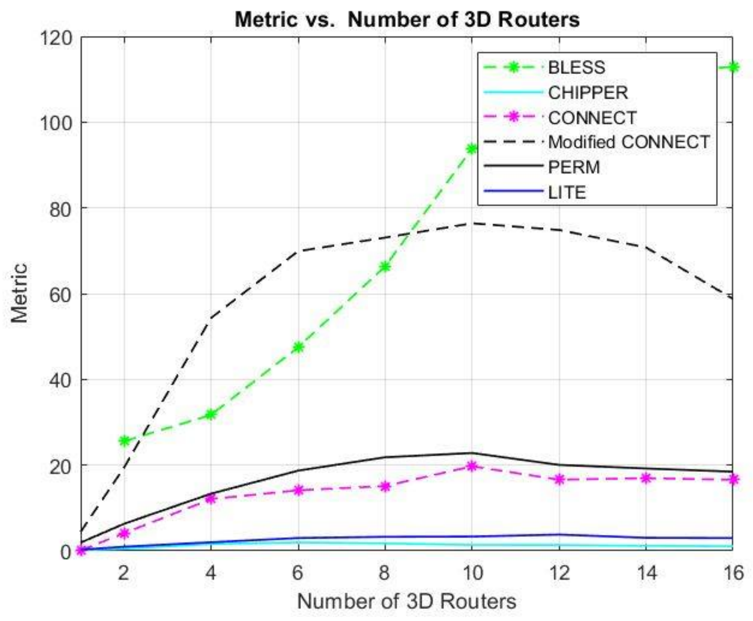


Fig. 5.3: Optimum Number of Three-Dimensional Routers

Modified CONNECT; otherwise, there is no correlation. The optimum number of 3D routers depends on the architecture of the router. The simulations are conducted in a 4x4x3 network using uniform traffic pattern. The optimum number of 3D routers for 3D BLESS is 16 routers. The optimum number of 3D routers for CONNECT and Modified CONNECT is 10 routers. The optimum number of 3D routers for PERM is 10 and 12 for Lite. This is illustrated in Fig.5.3. It is noticed that Modified CONNECT has the highest FOM. Taking the operating frequency into account, Modified CONNECT has FOM higher than BLESS and Lite.

The optimum location of 3D routers depends on the traffic pattern. Simulations are conducted using uniform traffic pattern to find the optimum locations of 3D routers. The simulations included CONNECT, Modified CONNECT, 3D-BLESS, 3D-PERM, and Lite. The 4x4x3 NoC of these routers are synthesized on Virtex 7 evaluation kit(xc7vx485tffg1761-2). The simulations are conducted with respect to the FOM. Below are the cases which are included in the simulations. The target was to find the locations of 3D routers that maximize the FOM, if the number of 3D routers in the network is 1, or 2, or 4, or 6 or 8. The simulation is repeated for each router design. The five routers obtain the same results. The router indices are ordered as shown in Fig.5.4.

- *Single 3D router:* The best location for a single 3D router is at the middle of network to boost the performance. The nodes at the middle of the network are the most congested ones. The optimum choice is router 5 as shown in Fig5.4.

- *Two 3D routers:* The best location is also at the middle of network, where the two routers are on the diagonal of the mesh. However, the performance is better using router 5 and router 10 as three-dimensional routers rather than using router 6 and router 9 as three-dimensional routers.

- *Four 3D routers:* The best location for the four 3D routers is at the middle of the network. The next best location is at the corners of the network, where router 3, router 12, router 0

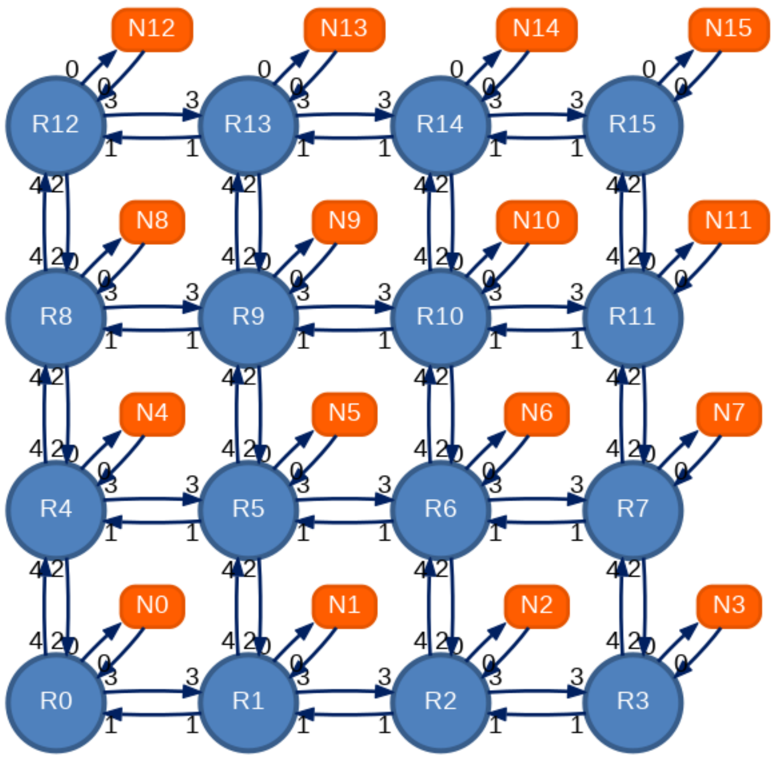


Fig. 5.4: Routers Indices in 4x4 Mesh Network[45]

and router 15 are the 3D routers.

- *Six 3D routers:* The best location is the four corner routers and the two on the diagonal especially router 5 and router 10.

Therefore, it is concluded that the locations depend on the traffic pattern. It is concluded also that 3D routers should be placed at the hotspots. Therefore, if the traffic is uniform, the optimum locations of the 3D routers is at the middle of the mesh.

5.2 CAD TOOL

A new CAD tool is proposed in Fig.5.5. The tool enables the user to configure the network, simulate it and easily synthesize it. The tool is developed to allow more experiments to be conducted on buffer-less 3D network. The tool allows the user to choose one of the buffer-less 3D routers 3D BLESS, 3D PERM, 3D CHIPPER and 3D Modified CONNECT. The routers are synthesized on Virtex 7. Modified CONNECT operates at 100 MHz, while the other four routers operate at 10 MHz.

The user can configure these parameters. 3D Nocet [14] uses one 3D router per mesh where the user could change the topology. The user could choose mesh or ring topology. The router uses elevator first algorithm.

●*Buffer-less Router.*

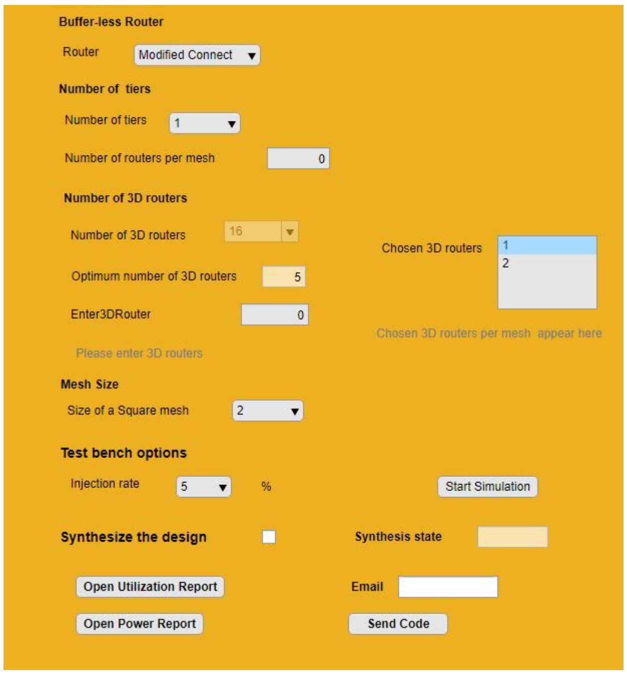
●*Mesh Size.*

●*Number of tiers.*

●*Number of 3D routers.*

●*3D Routers.*

●*Injection rate.*



Buffer-less Router

Router: Modified Connect

Number of tiers

Number of tiers: 1

Number of routers per mesh: 0

Number of 3D routers

Number of 3D routers: 16

Optimum number of 3D routers: 5

Enter3DRouter: 0

Please enter 3D routers

Chosen 3D routers: 1, 2

Chosen 3D routers per mesh appear here

Mesh Size

Size of a Square mesh: 2

Test bench options

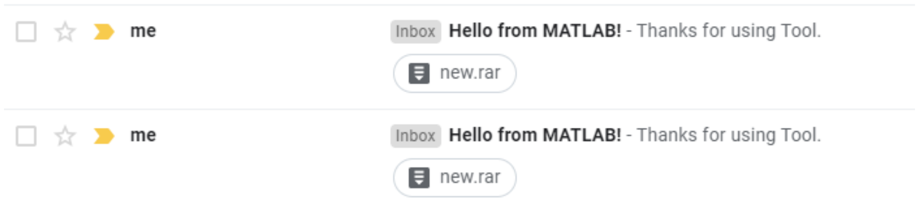
Injection rate: 5 %

Synthesize the design ☐

Synthesis state

Email

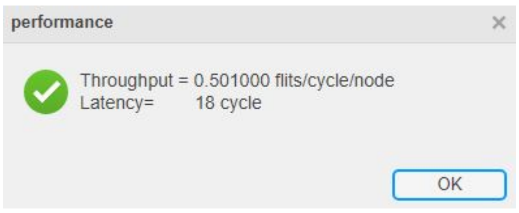
Fig. 5.5: The GUI of CAD Tool



me — Inbox **Hello from MATLAB!** - Thanks for using Tool.
new.rar

me — Inbox **Hello from MATLAB!** - Thanks for using Tool.
new.rar

Fig. 5.6: Network code received by mail



performance

Throughput = 0.501000 flits/cycle/node
Latency= 18 cycle

Fig. 5.7: Performance

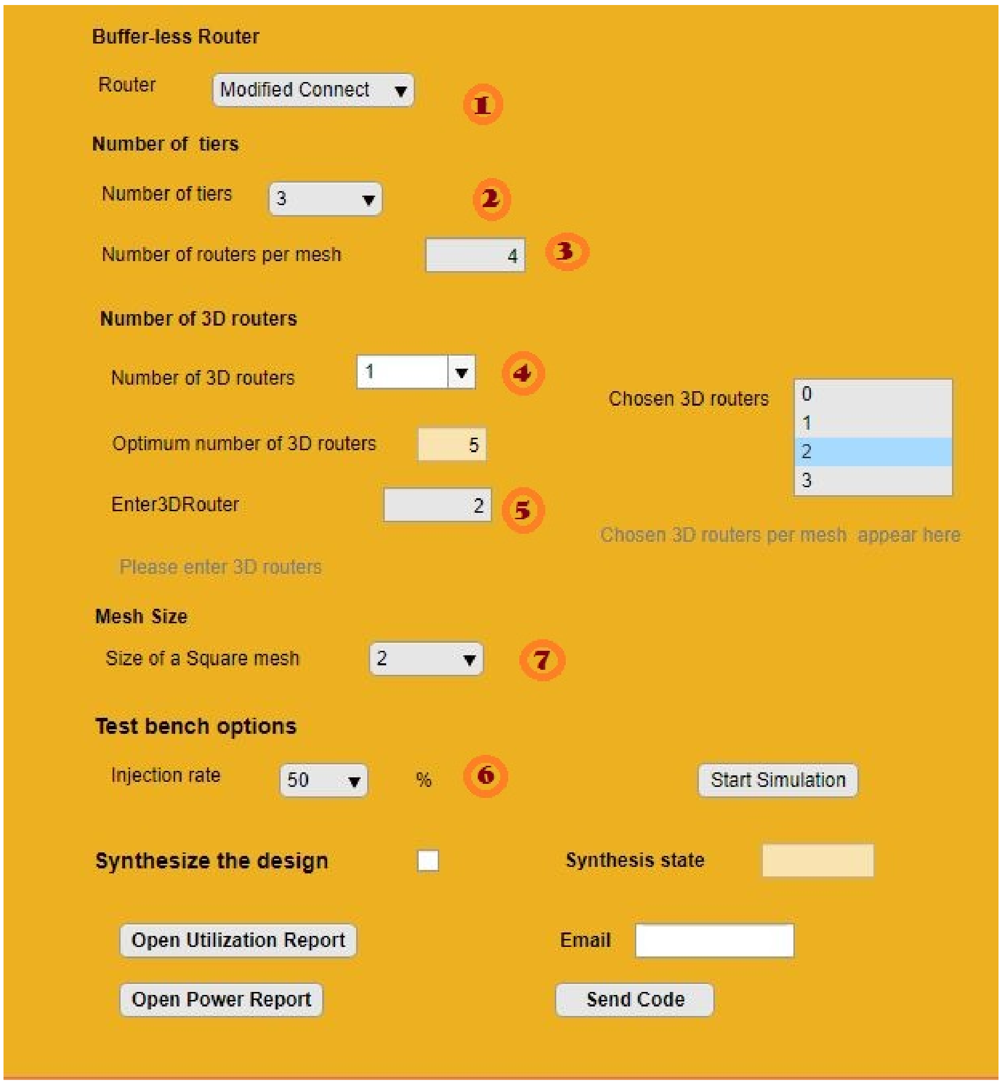
The tool uses mesh topology. The square mesh size could be changed from 2x2 to 12x12. The number of tiers varies from 1 to 8. The user could choose the number of 3D routers per mesh.

All the mesh layers are exactly identical. They have the same size and the same number of 3D routers. The 3D routers exist exactly at the same locations on each tier. The user can simulate the network. The tool uses uniform traffic pattern. However, the user could choose the average packet injection rate. To start simulation press “Start simulation” button. Once the network is simulated, the tool displays the average network latency and network throughput as shown in Fig.5.7. The user could choose one of these routers: Modified CONNECT, BLESS, Perm, Chipper and Lite. The generated code could be sent to the provided email address as an attachment 5.6. The user should disable the anti-virus protection to receive the attachment. The generated code is in Verilog. The user can synthesize the network and implement it. To synthesize the network press the “Synthesize the design” check box. Implementation starts automatically once the synthesis completes. While the network is being synthesized and implemented, “Synthesis state” displays “Running”. Once the network is implemented, “Done” is displayed. Later, the user could open the reports by pressing “Open Utilization Report” to open the area utilization report; and “Open Power Report” to open power report.

As the user chooses the buffer-less router, the tool displays a recommended number of 3D routers in “Optimum number of 3D routers”. The user could enter the number of 3D routers in “Number of 3D Routers”. Then the user could enter the index of 3D routers in “Enter3DRouter” edit field. To enter the index of 3D router, the user should type the index then press enter. If the user wants to add another router, omit the previous index, type the index and press enter. The chosen 3D routers appear at “Chosen 3D routers”. It is recommended to follow the order of buttons while configuring the network. The user should choose first the router, then the number of tiers, then the number of routers per mesh as shown by the numbers on Fig.5.8. The CAD tool is created using Matlab app designer unlike 3D-NOCET, it is created by Linux commands.

A walkthrough example

In the figure 5.8 the user has selected Modified CONNECT router. The user has selected 3 tiers. The single tier is 2x2 mesh; that is why ”Number of routers per tier” is 4 and ”size of the mesh” is 2. The user has selected one 3D router per tier. The 3D router is that of index 2. The “chosen 3D routers” displays all the possible router indices in a mesh layer and the 3D ones are shaded; therefore in this example the four indices 0,1,2,3 are displayed in “chosen 3D routers” and index 2 is shaded. The injection rate used in simulation is flit per cycle. The injection rate represents the rate of injection of packets which is 50%. As the packet consists of two flits.



Buffer-less Router

Router: Modified Connect

Number of tiers

Number of tiers: 3

Number of routers per mesh: 4

Number of 3D routers

Number of 3D routers: 1

Optimum number of 3D routers: 5

Enter3DRouter: 2

Please enter 3D routers

Chosen 3D routers: 0, 1, 2, 3

Chosen 3D routers per mesh appear here

Mesh Size

Size of a Square mesh: 2

Test bench options

Injection rate: 50 %

Synthesize the design

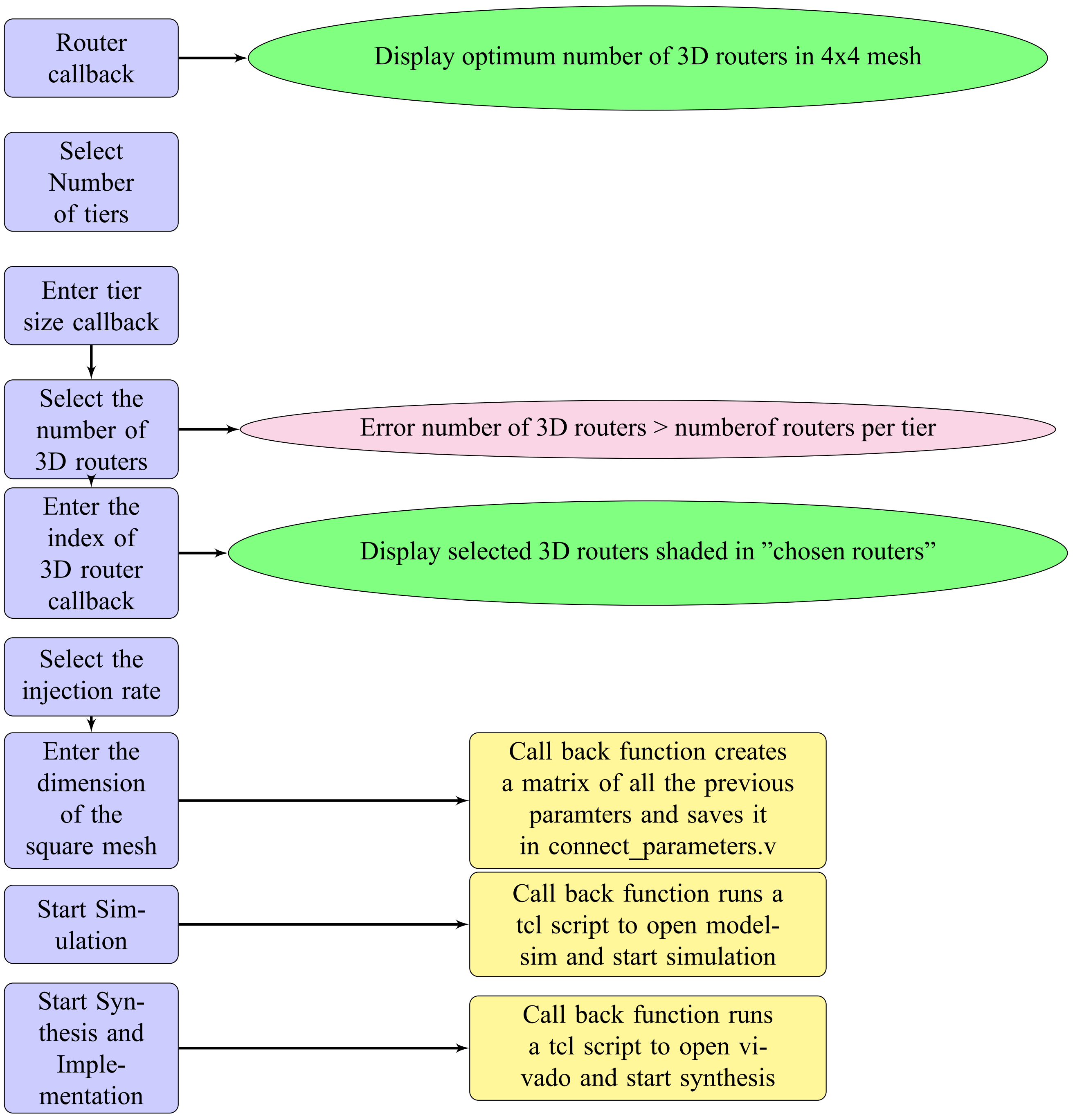
Synthesis state

Email

Fig. 5.8: Walkthrough example

5.2.1 The execution flow of the CAD Tool

This diagram illustrates the callback functions of the matlab tool and the execution flow of the tool and the execution of the tcl codes.


Router callback
Display optimum number of 3D routers in 4x4 mesh
Select Number of tiers
Enter tier size callback
Select the number of 3D routers
Error number of 3D routers > numberof routers per tier
Enter the index of 3D router callback
Display selected 3D routers shaded in ”chosen routers”
Select the injection rate
Enter the dimension of the square mesh
Call back function creates a matrix of all the previous paramters and saves it in connect_parameters.v
Start Simulation
Call back function runs a tcl script to open modelsim and start simulation
Start Synthesis and Implementation
Call back function runs a tcl script to open vivado and start synthesis

This tool is designed to run 3D buffer-less network with generic parameters. The network could have been created using matlab and implemented using HDL coder, however HDL coder creates RTL designs with poor specs. Therefore, the proposed tool simulate and synthesize RTL networks. it does not generate any RTL blocks.

The callback functions of the buttons and the edit fields can be found in Appendix A. This tool edit the generic parameters in verilog parameters file. Each button edit a certain parameter in a matrix of generic parameters at the end "Size of a Square mesh" uploads this matrix in verilog parameter file. This is why "Size of a Square mesh" should be the last parameter that the user could edit in the network before synthesis or simulation. This tool allows the user to perform behavioral simulation on modelsim, it does not allow the user to perform post-synthesis simulation. The callback of "Start Simulation" opens a tcl script which compiles the verilog design and simulate it. However, Matlab has this command that can execute more than one tcl instruction.

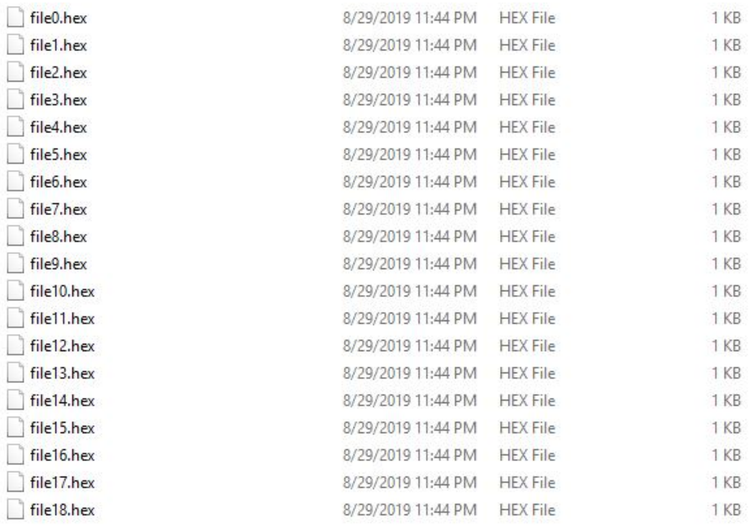
```
vsim('tclstart',{'vlog *.v','vsim work.CONNECT_testbench_sample','run -all'});
```

After the simulation the results (performance and latency) are written in a txt file that matlab opens and displays as in Fig.5.7.

The callback function of "Size of a Square mesh":
If the chosen router is CONNECT or Modified CONNECT, this callback function creates the routing tables of each router in the network. The tool creates routing tables equal to the number of routers in the network. Each routing table has number of entries equal to the number of routers in the network. In each entry is saved the port number to reach the corresponding destination. The destination is the address number in this RAM. Attached in APPENDIX A, a matlab function that compares the current router with every possible destination.
If the destination is on the left, it is given port 1.
If the destination is on the right, it is given port 3.
If the destination is on the top, it is given port 4.
If the destination is on the bottom, it is given port 2.
If the destination is on another tier, it is routed to the nearest 3D router.
If the current router is a 3D router & the destination is in a tier above the current one, it is given port 5.
If the current router is a 3D router & the destination is in a tier below the current one, it is given port 6.

Attached in Appendix A, a function that calculates the nearest 3D router to the given router. The function receives an array of the indices of all the 3D routers per mesh and the index of the current router. It calculates the number of hops separating the current router and each of them. Then it returns the nearest one.

Figure 5.9 shows the routing tables created by the tool. These routing tables are created for CONNECT or Modified CONNECT only. These routing tables are implemented as distributed RAMs.



file0.hex	8/29/2019 11:44 PM	HEX File	1 KB
file1.hex	8/29/2019 11:44 PM	HEX File	1 KB
file2.hex	8/29/2019 11:44 PM	HEX File	1 KB
file3.hex	8/29/2019 11:44 PM	HEX File	1 KB
file4.hex	8/29/2019 11:44 PM	HEX File	1 KB
file5.hex	8/29/2019 11:44 PM	HEX File	1 KB
file6.hex	8/29/2019 11:44 PM	HEX File	1 KB
file7.hex	8/29/2019 11:44 PM	HEX File	1 KB
file8.hex	8/29/2019 11:44 PM	HEX File	1 KB
file9.hex	8/29/2019 11:44 PM	HEX File	1 KB
file10.hex	8/29/2019 11:44 PM	HEX File	1 KB
file11.hex	8/29/2019 11:44 PM	HEX File	1 KB
file12.hex	8/29/2019 11:44 PM	HEX File	1 KB
file13.hex	8/29/2019 11:44 PM	HEX File	1 KB
file14.hex	8/29/2019 11:44 PM	HEX File	1 KB
file15.hex	8/29/2019 11:44 PM	HEX File	1 KB
file16.hex	8/29/2019 11:44 PM	HEX File	1 KB
file17.hex	8/29/2019 11:44 PM	HEX File	1 KB
file18.hex	8/29/2019 11:44 PM	HEX File	1 KB

Fig. 5.9: Routing tables Created by the CAD Tool

This tool uses two projects, the first one when it compiles CONNECT or Modified CONNECT; the other when it compiles 3D-PERM, 3D-BLESS, 3D-CHIPPER or 3DAPBLESS. In order to specify which router is the 3D router and which is not, a stream of bits is created by the tool, the number of bits equal to the number of routers. 1 is denotes to 3D router and 0 denotes toa 2D router. This stream of bits is converted into decimal and saved in the parameter file. RTL deals with this decimal number as an array of signals.

In order to display the "Chosen 3D routers", a function is available in APPENDIX A. This function is the callback of "Enter3DRouter". An array is created if the number of tiers is greater than one. This array has the indices of the 3D routers per the first tier from 1 to N. Every time the user enter the index of a new router, the index is incremented and the value is shaded.

Recommendations for Future Work

- Discuss a new mechanism in assembly buffers.
  Chipper is the only design that addressed assembly buffers problem. As long as each flit in the packet is routed independently, flits will reach the destination nodes out of order. The router should have a large buffer to assemble the packets. There is a cost penalty to this buffer-less schemes. It is more efficient to force flits of the same packet to use the same path.

- Design a 3D buffer-less FPGA where the 3D routers are hardwired.
  The performance of the system depends on the location of the 3D routers. It is better to place the 3D routers at the hotspots. Therefore, this FPGA would probably be suitable for a specific application and with a specific traffic pattern.

- Design a new FPGA that will use dynamic partial reconfiguration (DPR) with the buffer-less routers in order to use the FPGA properly according to the running application. The challenge is to make all the buffer-less routers have the same packet format and the same interface.

- Discuss another topology that could help reduce the area occupied by the switch. In order to reduce the complexity of the switch, the switch should not allow the full mapping of the input ports to the output ports. Changing the number of ports and the network connectivity would help decrease the set of combination inside the switch. There is a trade off between the performance and the area of the switch. Also, livelock and deadlock will be critical concerns in the design. This approach is used in CHIPPER and it reduces the permutation network from three stages to two stages only.
- Apply a generic algorithm not a heursitic one to optimize the figure of merit with respect to the parameters.

Discussion and Conclusion

Interconnects occupy significant area of on chip networks. Interconnects control the throughput of the network especially for new technologies. Buffer-less NoCs have been proposed as an alternative to buffered NoCs tackling the high network power head-on by eliminating buffers. Buffer-less routers are efficient at low and medium traffic loads; however they have low saturation throughput.
The main contribution of this work is listed as follows:

- Introduction of a new efficient lightweight buffer-less router ”Modified CONNECT”. This work is published in [47].
- Introduction of Modified CONNECT in 3D networks.
- Investigation of 3D buffer-less network parameters.
- Introduction of a new CAD Tool that enables the user to perform more experiments on 3D buffer-less networks.

In this thesis, A brief introduction on networks on chip is discussed in Chapter1. Chapter2 illustrates the conventional buffered router on chip; it illustrates the design and the architecture of CONNECT. A short survey follows in Chapter 3. The survey includes the related work in buffer-less routers. The Survey includes BLESS, Chipper, Minbd, Scepter, Carpool, Slider, Cascade and QBLESS. Chapter 3 includes also the open source simulators Noculator, Booksim and Notsrum. Chapter 4 describes the contribution of this work. It illustrates the architecture of Modified CONNECT; It also includes an introduction to 3D NoCs. It describes the fabrication of 3D NoCs and the 3D routing techniques. Also, it describes the pros and cons of 3D NoCs. It presents 3D Modified CONNECT architecture. 3D Modified CONNECT is evaluated in 3D NoCs against the other buffer-less designs in Chapter 4 as well. In Chapter 5 there is a brief study on the optimum size of the 3D NoCs. The optimum number of layers, the optimum size of mesh and the optimum number of 3D routers per mesh and their locations. Finally, Chapter 5 presents a CAD tool that summarize this work. The user could choose the buffer-less router, the size of mesh, the number of layers, the number of 3D routers and their locations. The tool could synthesize and simulate the network. The code could be sent to the provided email address.

These are the main results that could be concluded:

- Modified CONNECT has the minimal cost, where it achieves 30% reduction in area compared to CONNECT, 24% compared to BLESS and 18% compared to CHIPPER.
- Modified CONNECT achieves a comparative performance to the buffered original CONNECT.

- Modified CONNECT is suitable for wide datapath data.

- Modified CONNECT has small area because the router uses the routing tables instead of routing functions; and because it is a single stage pipeline.

- Three-dimensional network might be a reasonable solution to network performance limits in NoC. The performance is degraded by increasing network diameter much more than by increasing the number of layers.

- 3D-Modified CONNECT occupies half the area consumed by 3DAPBLESS. 3D-Modified CONNECT outperforms 3D-Perm.

- The switch occupies the largest area by the router on FPGA.

- The topology can decrease the complexity of the switch significantly. This is used in Chipper and Hoplite.

- The optimum number of 3D routers in a network actually depends on the architecture of the router.

- The optimum number of 3D routers in a 4x4 mesh for BLESS, Chipper, Perm, 3DAPBLESS, CONNECT and Modified CONNECT is 16, 6, 10, 12, 10 and 10 respectively.

- As the size of the mesh increases, the FOM decreases. Details are included in Chapter 5.

- As the number of layers increases, the FOM 3D NoC decreases. Details are included in Chapter 5.

Bibliography

[1] http://www.monolithic3d.com/blog/fpga-as-asic-alternative-past-and-future,2020.

[2] http://www.eng.ucy.ac.cy/theocharides/Courses/ECE408/7%20Series%20FPGA%20Overview.pdf,2020.

[3] https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01016.pdf,2020.

[4] https://ece757.ece.wisc.edu/lect09-interconnects-5-microarchitecture.pdf,2020.

[5] https://www.open-silicon.com/2-5d-technology/,2020.

[6] http://parallelcomp.github.io/Lecture3.pdf,2020.

[7] http://srbiau.ac.ir/Files/booksim_manual.pdf,2020.

[8] http://users.encs.concordia.ca/~asim/COEN_6501/Lecture_Notes/FPGA%20Report.pdf,2020.

[9] https://anysilicon.com/flipchip-package-overview/,2020.

[10] FPGA vs. ASIC, what to choose? https://anysilicon.com/fpga-vs-asic-choose/,2020.

[11] When will FPGAs kill ASICs. https://www.doc.ic.ac.uk/~wl/teachlocal/arch/killasic.pdf,2020.

[12] M. S. Abdelfattah and V. Betz. Networks-on-chip for FPGAs: hard, soft or mixed? In *TRETS*, 2014.

[13] D. U. Becker. *Efficient microarchitecture for network-on-chip routers*. PhD thesis, Stanford University, 2012.

[14] M. Beheiry, H. Mostafa, Y. Ismail, and A. M. Soliman. 3D-NOCET: A tool for implementing 3D-NoCs based on the direct-elevator algorithm. In *ISQED*, pages 144–148, 2017.

[15] E. Bolotin, I. Cidon, and R. Ginosar. Cost consideration in Network on Chip. *Integration VLSI Journal*, pages 19–42, 2004.

[16] C.Feng, Z. Lu, A. Jantsch, and M. Zhang. A 1-cycle 1.25GHz bufferless router for 3D Network-on-Chip. *IEICE*, pages 1519–1522, 2012.

[17] B. K. Daya, L.-S.Peh, and A. P. Chandrakasan. Quest for high-performance bufferless NoCs with single-cycle express paths and self learning throttling. In *DAC*, pages 36:1–36:6, 2016.

[18] H. Elmiligi, A. Morgan, and et al. Power optimization for application-specific Networks on Chips: a topology-based approach. *Microprocessors and Microsystems 33*, pages 343–355, 2009.

[19] B. Nayak et al. SLIDER: Smart late injection deflection router for mesh NoCs. In *ICCD*, pages 377–383, 2013.

[20] C. Fallin et al. CHIPPER: A low complexity buffer-less deflection router. In *HPCA*, pages 144–155, 2011.

[21] C. Fallin et al. MinBD: Minimally-buffered deflection routing for energy-efficient interconnect. In *NOCS*, pages 1–10, 2012. `https://github.com/CMU-SAFARI/NOCulator`, 2020.

[22] G. Nychis et al. Next generation on-chip networks: What kind of congestion control do we need? In *Hotnets-IX*, 2010.

[23] H. A. H. Fahmy et al. Design guidelines for soft implementations to embedded NoCs of FPGAs. In *IDT*, pages 37–42, 2016.

[24] H. A. H. Fahmy et al. Exploiting the dynamic partial reconfiguration on NoC-based FPGA. In *NGCAS*, pages 277–280, 2017.

[25] J. Jose et al. DeBAR: Deflection based adaptive router with minimal buffering. In *DATE*, pages 1583–1588, 2013.

[26] J. Rabaey et al. *Digital Integrated Circuits*. 2002.

[27] Nan Jiang et al. A detailed and flexible cycle-accurate Network-on-Chip simulator. In *ISPASS*, 2013.

[28] O. Multu et al. Design and evaluation of hierarchical rings with deflection routing. In *SBAC-PAD*, 2014.

[29] O. Mutlu et al. On-chip networks from a networking perspective: congestion and scalability in many-core interconnects. In *SIGCOMM*, page 407–418, 2012.

[30] O. Mutlu et al. Carpool: A buffer-less on-chip network supporting adaptive multicast and hotspot alleviation. In *ICS*, pages 1–11, 2017.

[31] S. Attia et al. Comparative review of NoCs in the context of ASICs and FPGAs. In *ISCAS*, 2015.

[32] W.Davis et al. Demystifying 3D ICs : The pros and cons of going vertical. *IEEE*, pages 498–510, 2005.

[33] B. Feero and P. P. Pande. Performance evaluation for 3D Networks-on-Chip. In *ISVLSI*, pages 305–310, 2007.

[34] N. Gamal and H. Mostafa. Design guidelines for embeded NoCs on FPGA. In *ISQED*, pages 69–74, 2012.

[35] A. et al. Hassan. NoC-DPR: A new simulation tool exploiting the dynamic partial reconfiguration(DPR) on Network-on-Chip (NoC) Based FPGA. *Elsevier Integration VLSI Journal*, pages 204–212, 2018.

[36] C. Hu. *Modern Semiconductor Devices for Integrated Circuits*. 2009.

[37] C. Hu. *Semiconductor Devices for Integrated Circuits*. 2009.

[38] R. Jayaraman. (When) will FPGAs kill ASICs. In *DAC*, pages 321–322, 2001.

[39] G.R. Jonna, V.M. Thuniki, and M. Mutyam. CASCADE: Congestion aware switchable cycle adaptive deflection router. In *ARCS*, pages 35–47, 2016.

[40] N. Kapre and J. Gray. Hoplite: Building austere overlay NoCs for FPGAs. In *FPL*, pages 1–8, 2015. http://fpga.org/hoplite/, 2020.

[41] T. Kavitha, S. Shiyamala, and P. Nagarajan. FPGA implementation of arbiters algorithm for Network on Chip. *ARPN*, pages 11451–11456, 2016.

[42] N. H. Khan, S. M. Alam, and S. Hassoun. System-level comparison of power delivery design for 2D and 3D ICs. In *IEEE*, pages 1–7, 2009.

[43] P. Lajevardi. Design of a 3-dimension FPGA, 2005.

[44] T. Moscibroda and O. Mutlu. A case for buffer-less routing in on-chip networks. In *ISCA*, pages 196–207, 2009.

[45] M. K. Papamichael and J. C. Hoe. CONNECT: Re-examining conventional wisdom for designing NoCs in the context of FPGAs. In *FPGA*, pages 37–46, 2012. http://users.ece.cmu.edu/~mpapamic/connect/, 2020.

[46] V. F. Pavlidis and E. G. Friedman. 3-D topologies for Networks-on Chip. In *VLSI*, 2007.

[47] M. Shaheen, H. A. H. Fahmy, and H. Mostafa. Modified Connect: New buffer-less router for NoC-based FPGAs. In *MWSCAS*, pages 424–427, 2018.

[48] Chao-Ton Su and Tai-Lin Chiang. Optimal design for a ball grid array wire bonding process using a neuro-genetic approach. In *IEEE*, pages 13–18, 2002.

[49] K. Tatas. High-performance 3D NoC buffer-less router with approximate priority comparison. *MOCAST*, 2018.

AppendixA

A.1 TCL Command

The tcl commands used to synthesize the network RTL by Vivado.

```
open_project E:/simulink_work/type/project_1/project_1.xpr
reset_run synth_1
launch_runs synth_1
wait_on_run synth_1

launch_runs impl_1
wait_on_run impl_1

open_run impl_1

set_switching_activity -deassert_resets
report_utilization -file report1
report_power -name {power_1}

exit vivado
```

The tcl command used to simulate the network RTL by modelsim.

```
project open E:/simulink_work/typ/project1
vlog *.v
vsim work.testbench
run -all
```

A.2 Matlab

The callback of Simulate button
Once the simulation completes, the message in Fig.5.7 appears.

```
if strcmp(app.RouterDropDown.Value,'Modified Connect')
             cd ('E:\simulink_work\mc2')
           % vsim('tclstart',{'vlog *.v','vsim
              work.CONNECT_testbench_sample','run -all'});
             system('vsim -c -do E:/simulink_work/mc2/model.tcl');
             data = importdata('E:\simulink_work\mc2\Results','\n');
             message = sprintf('%s \n',data{:});
             f2 = uifigure;
             uialert(f2,message,'performance','Icon','success');
else
          cd ('E:\simulink_work\typ')
```

```
            % vsim('tclstart',{'vlog *.v','vsim work.testbench','run
                -all'},'runmode','Batch');
            system('vsim -c -do E:/simulink_work/typ/model.tcl');
            data = importdata('E:\simulink_work\typ\Results','\n');
            message = sprintf('%s \n',data{:});
            f = uifigure;
            uialert(f,message,'performance','Icon','success');
end
```

Below is the code used to configure Modified CONNECT network and change the design parameters in connect parameters.v.

```
M = importdata('E:\simulink_work\mc2\connect_parameters.v','`');
         M(1,1) = {['`define inj_rate',' ',app.InjectionrateDropDown.Value]
             };
         M(2,1) = {['`define Up',' ',linkD] };
         M(3,1) = {['`define Down',' ',linkD] };
         M(5,1) = {['`define Tier_Num',' ',app.NumberoftiersDropDown.Value]
             };
         M(7,1) = {['`define Mesh_Square','
             ',app.SizeofaSquaremeshDropDown.Value] };
         M(6,1) = {['`define Mesh_Size',' ',mesh_Num] };
         M(10,1) = {['`define dest_bits',' ',dest_str] };
         M(11,1) = {['`define High',' ',Routers_Num] };
         M(8,1) = {['`define addr_bits 68:',addr_str]};
         M(12,1) = {['`define FLIT_DATA_WIDTH',' ',flit_width_str]};
         filePh = fopen('E:\simulink_work\mc2\connect_parameters.v','w');
         fprintf(filePh,'%s\n',M{:});
         fclose(filePh);
```

The distributed RAMs are updated for Modified CONNECT.

```
layer= floor((myrouter-1)/Numpertier);
                   Rout3D = GetNearestRouter(app,myrouter-1,size);
                   R3D = ((Rout3D) + (layer*Numpertier));
                   if((myrouter-1 == R3D)&&
                       (floor((dest-1)/Numpertier)>floor((myrouter-1)/Numpertier)))
                         F(dest,myrouter)=5;
                     elseif((myrouter-1 == R3D)&&
                         (floor((dest-1)/Numpertier)<floor((myrouter-1)/Numpertier)))
                         F(dest,myrouter)=6;
                     elseif(mod(myrouter-1,size)<mod(R3D,size))
                         F(dest,myrouter)=3;
                     elseif(mod(myrouter-1,size)>mod(R3D,size))
                         F(dest,myrouter)=1;
                     elseif((mod(myrouter-1,size)==mod(R3D,size))&&
                         (R3D)>(myrouter-1))
                         F(dest,myrouter)=4;
                     else
                         F(dest,myrouter)=2;
                     end
```

A function to get the nearest three dimensional router.

```
function nearest3Rout = GetNearestRouter(app,mypos,mesh)
          % datarr = getappdata(routers3,Dim3);
          datarr=app.router3D;
           msize = mesh*mesh;
           i= int16(mod(mypos,mesh));
           j= int16(floor(mod(mypos,msize)/mesh));
           %sz = repmat(mypos,1,str2num(app.Numberof3DroutersDropDown.Value));
           J = int16(floor(datarr/mesh));
           I = int16(mod(datarr,mesh));
           hop = abs(I-i)+ abs(J-j);
           [M ,Index] = min(hop);
           nearest3Rout = datarr(Index);
       end
```

The callback function of Send Code button

```
 UserName = 'mro.isp@gmail.com';
          passWord = 'greenblue';
          setpref('Internet','E_mail',UserName);
          setpref('Internet','SMTP_Server','smtp.gmail.com');
          setpref('Internet','SMTP_Username',UserName);
          setpref('Internet','SMTP_Password',passWord);
          props = java.lang.System.getProperties;
          props.setProperty('mail.smtp.auth','true');
          props.setProperty('mail.smtp.socketFactory.class', ...
                 'javax.net.ssl.SSLSocketFactory');
          props.setProperty('mail.smtp.socketFactory.port','465');

        if strcmp(app.RouterDropDown.Value,'Modified Connect')

         if isfile('E:\simulink_work\new')
          cd('E:\simulink_work');
          delete('new');
         end
          cd('C:\');
          system('"Program Files\WinRaR\WinRaR.exe" a
              -x"E:\simulink_work\mc2\project_1"
              -x"E:\simulink_work\mc2\project.mpf"
              -x"E:\simulink_work\mc2\project.cr.mti"
              -x"E:\simulink_work\mc2\vsim.wlf" "E:\simulink_work\new"
              "E:\simulink_work\mc2\"');
        sendmail(app.EmailEditField.Value,'Hello from MATLAB!','Thanks for
            using Tool.','E:\simulink_work\new.rar');
        else


          if isfile('E:\simulink_work\newtyp.rar')
            %cd('E:\simulink_work');
          delete('E:\simulink_work\newtyp.rar');
         end
```

```
          cd('C:\');
          system('"Program Files\WinRaR\WinRaR.exe" a
              -x"E:\simulink_work\typ\project_1"
              -x"E:\simulink_work\typ\project1.mpf"
              -x"E:\simulink_work\typ\project.cr.mti"
              -x"E:\simulink_work\typ\vsim.wlf" "E:\simulink_work\newtyp"
              "E:\simulink_work\typ\"');
        sendmail(app.EmailEditField.Value,'Hello from MATLAB!','Thanks for
            using Tool.','E:\simulink_work\newtyp.rar');

        end
```

The callback function of the list box "Enter3DRouter"

```
app.router3D(app.index3D) = value;
app.index3D = app.index3D +1;
app.Chosen3DroutersListBox.Value = string(app.router3D);
```

A.3 Verilog

The small block in the Deflection Unit of Modified CONNECT allocator

This block is connected in a matrix form as shown in Fig.4.2.

```
`ifdef BSV_ASSIGNMENT_DELAY
`else
`define BSV_ASSIGNMENT_DELAY
`endif

module grantport(CLK,
             RST_N,
             notgranted,
             notlocked,
              grant,
              carryhorizontal,
              carryvertical
             );

input CLK;
input RST_N;

input notgranted;
input notlocked;

output grant;
output carryhorizontal;
output carryvertical;


wire grant;
wire carryhorizontal;
wire carryvertical;

assign grant = notgranted && notlocked;

assign carryhorizontal = notgranted && !notlocked;

assign carryvertical = !notgranted && notlocked;

endmodule
```

Another way to implement the deflection block is to use these three successive stages

```
module outputarbiter(CLK,
              RST_N,
              notgrant1,
              notgrant2,
              notgrant3,
              notgrant4,
              output_1_select,
              EN_output_arbs_1_next,
              output_2_select,
           EN_output_arbs_2_next,
              output_3_select,
           EN_output_arbs_3_next,
           output_4_select,
           EN_output_arbs_4_next);

  input CLK;
  input RST_N;

  input EN_output_arbs_1_next;
  input EN_output_arbs_2_next;
  input EN_output_arbs_3_next;
  input EN_output_arbs_4_next;

  input notgrant1, notgrant2, notgrant3, notgrant4;

  output [4:0] output_1_select,
         output_2_select,
         output_3_select,
         output_4_select;

  // signals for module outputs
  wire [4 : 0] output_1_select,
         output_2_select,
         output_3_select,
         output_4_select;

   wire notgrantpA, notgrantpB, notgrantpC, notgrantpD;
   wire [4:0] forport1or2A, forport3or4A, forport3or4B, forport1or2B;
   wire [3:0] inforport1or2, inforport3or4,inforport3or4B, inforport1or2B;
   wire [1:0] numout1, numout2, numout3, numout4;
   wire [1:0] numout3or4, numout1or2, numout3or4b, numout1or2b;
   wire [1:0] port1num, port2num, port3num, port4num;
   wire EN1, EN2, EN3, EN4;

stage2 stC(.notgrant1(notgrant1),
             .notgrant2(notgrant2),
             .EN1(!EN1),
             .EN2(!EN2),
             .num1(2'd1),
             .num2(2'd2),
             .numA(numout1),
             .numB(numout2),
             .defnum(numout3or4),
             .defnum1(numout3or4b));

stage2 stD(.notgrant1(notgrant3),
          .notgrant2(notgrant4),
          .EN1(!EN3),
             .EN2(!EN4),
             .num1(2'd1),
             .num2(2'd2),
             .numA(numout3),
             .numB(numout4),
             .defnum(numout1or2),
             .defnum1(numout1or2b));

      stage3 sTE(.num1(numout3or4),
               .num2(numout3or4b),
               .notgrant1(|numout3or4),
               .notgrant2(|numout3or4b),
                .EN1(!EN3 && !(|numout3)),
                .EN2(!EN4 && !(|numout4)),
                .numA(port3num),
                .numB(port4num));

        stage3 sTF(.num1(numout1or2),
                 .num2(numout1or2b),
                 .notgrant1(|numout1or2),
                 .notgrant2(|numout1or2b),
                .EN1(!EN1&& !(|numout1)),
                .EN2(!EN2 && !(|numout2)),
                .numA(port1num),
                .numB(port2num));

   assign output_1_select= {port1num,numout1,1'd0};
   assign output_2_select= {port2num,numout2,1'd0};
   assign output_3_select= {numout3,port3num,1'd0};
   assign output_4_select= {numout4,port4num,1'd0};
```

```
  assign EN1 = EN_output_arbs_1_next ;
  assign EN2 = EN_output_arbs_2_next ;
  assign EN3 = EN_output_arbs_3_next ;
  assign EN4 = EN_output_arbs_4_next ;

endmodule // mkRouterOutputArbitersStatic
```

Stage 1

```
`ifdef BSV_ASSIGNMENT_DELAY
`else
`define BSV_ASSIGNMENT_DELAY
`endif

module stage1(num1,
             num2,
             notgrant1,
             notgrant2,
             Enable1,
             Enable2,
             numA,
             numB,
             notgrantA,
             notgrantB);

  // value method gen_grant_carry
  input Enable1;
  input Enable2;
  input [3:0] num1, num2;
  input notgrant1, notgrant2;
  output [3:0] numA, numB;
  output notgrantA, notgrantB;

  wire [3:0] num1, num2;

  // value method gen_grant_carry
  assign numA= notgrant1 &&(Enable1 || Enable2)? num1: num2;
  assign numB= notgrant1 &&(Enable1 || Enable2)? num2: num1;

  assign notgrantA = ( notgrant1 &&(Enable1 || Enable2))? notgrant1: notgrant2;
  assign notgrantB = ( notgrant1 &&(Enable1 || Enable2))? notgrant2: notgrant1;
endmodule // module_gen_grant_carry
```

Stage2

```
`ifdef BSV_ASSIGNMENT_DELAY
`else
`define BSV_ASSIGNMENT_DELAY
`endif
```

```
module stage2(notgrant1,
            notgrant2,
            num1,
            num2,
             EN1,
             EN2,
             numA,
             numB,
             defnum,
             defnum1);

  // value method gen_grant_carry
  input EN1,EN2;
  input num1, num2;
  output [1:0] numA, numB;
  output [1:0] defnum, defnum1;
  input notgrant1, notgrant2;
  wire [1:0] num1, num2;

  assign numA = {EN1&& notgrant2 && !notgrant1 ,
                EN1 && notgrant1};
  assign numB = {!(!EN1 && notgrant1 && EN2)&&
                ((!EN1 && notgrant2 && EN2) ||
                 (EN1 && EN2 && notgrant1 &&notgrant2))
                ,!EN1 && notgrant1 && EN2};

  assign defnum= (!EN1&& !EN2 && notgrant1)? num1:
                (!EN1&& !EN2 && notgrant2)? num2:
                (!EN1 && EN2 && notgrant1 && notgrant2)? num2:
                (EN1 && !EN2 && notgrant1 && notgrant2)? num2:2'd0;

  assign defnum1 = (!EN1&& !EN2 && notgrant2 && notgrant1)? num2: 2'd0;

endmodule
```

Stage3

```
`ifdef BSV_ASSIGNMENT_DELAY
`else
`define BSV_ASSIGNMENT_DELAY
`endif

module stage3(notgrant1,
            notgrant2,
            num1,
            num2,
             EN1,
             EN2,
             numA,
             numB);
```

```
  // value method gen_grant_carry
  input EN1,EN2;
  input num1, num2;
  output [1:0] numA, numB;
  input notgrant1, notgrant2;
  wire [1:0] num1, num2;

   assign numA= (EN1 && notgrant1)? num1: (EN1 && notgrant2)?num2 : 2'd0;
  assign numB= (!EN1 && notgrant1 && EN2)? num1 : (!EN1 && notgrant2 && EN2)?
      num2:(EN1 && EN2 && notgrant1 &&notgrant2)?num2:2'd0;


     endmodule
```

2D BLESS

```
`include "defines.v"

module brouter_bless2D
#(parameter   addr = 6'b000101,
  parameter   noc = 0)
(
    input      `control_w port0_ci,
    input      `control_w port1_ci,
    input      `control_w port2_ci,
    input      `control_w port3_ci,
    input      `control_w port4_ci,
    input      `data_w    port0_di,
    input      `data_w    port1_di,
    input      `data_w    port2_di,
    input      `data_w    port3_di,
    input      `data_w    port4_di,
    input                 clk,
    input                 rst,
    output     `control_w port0_co,
    output     `control_w port1_co,
    output     `control_w port2_co,
    output     `control_w port3_co,
    output     `control_w port4_co,
    output     `data_w    port0_do,
    output     `data_w    port1_do,
    output     `data_w    port2_do,
    output     `data_w    port3_do,
    output     `data_w    port4_do,
    output                port4_ready);

    // Config
    wire   `addrx_w  addrx, max_addrx;
    wire   `addry_w  addry, max_addry;
    wire   `addrz_w  addrz, max_addrz;
    wire   [`Mesh_Size -1:0] enable_Dim;
```

```
assign addrx = addr[`addrx_f]; // This nodes x address
assign addry = `Mesh_Square -1 - addr[`addry_f]; // This nodes y address
assign addrz = addr[`addrz_f]; // This nodes y address
assign max_addrx = `addrx_max;
assign max_addry = `addry_max;
assign max_addrz = `addrz_max;

// Input wires for reset
wire   `control_w port0_cin, port1_cin, port2_cin, port3_cin, port4_cin;

assign port0_cin = (rst) ? `control_n'd0 : port0_ci;
assign port1_cin = (rst) ? `control_n'd0 : port1_ci;
assign port2_cin = (rst) ? `control_n'd0 : port2_ci;
assign port3_cin = (rst) ? `control_n'd0 : port3_ci;
assign port4_cin = (rst) ? `control_n'd0 : port4_ci;

// Resource Ready Wires
wire   all_valid;
wire   resource_go0, resource_go1, resource_go2, resource_go3;
wire   validin;
wire   port4_read;

// Cross Stage Wires
wire   `control_w port0_c1, port1_c1, port2_c1, port3_c1, port4_c1;
wire   `data_w    port0_d1, port1_d1, port2_d1, port3_d1, port4_d1;
wire   `control_w port0_c2, port1_c2, port2_c2, port3_c2, port4_c2;
wire   `data_w    port0_d2, port1_d2, port2_d2, port3_d2, port4_d2;

// Routing Matrices
wire   `rmatrix_2w rmatrix0, rmatrix1, rmatrix2, rmatrix3, rmatrix4;

// Final Route
wire   `routecfg_2w route_config;

reg port_ready_d;

/************* STAGE 1 *************/
reg `control_w port0_r, port1_r, port2_r, port3_r, port4_r;
always @(posedge clk) begin
    port0_r <= port0_cin;
    port1_r <= port1_cin;
    port2_r <= port2_cin;
    port3_r <= port3_cin;
    port4_r <= port4_cin;
end

   assign enable_Dim = `UP;
// Route Computation
route_bless2D rc0(.control_in(port0_r),
                 .enable(enable_Dim),
                 .addrx(addrx),
               .addry(addry),
               .addrz(addrz),
               .addrx_max(max_addrx),
               .addry_max(max_addry),
               .addrz_max(max_addrz),
               .clk(clk),
               .rst(rst),
               .resource_go(resource_go0),
               .rmatrix(rmatrix0));

route_bless2D rc1(.control_in(port1_r),
               .enable(enable_Dim),
               .addrx(addrx),
               .addry(addry),
               .addrz(addrz),
               .addrx_max(max_addrx),
               .addry_max(max_addry),
               .addrz_max(max_addrz),
               .clk(clk),
               .rst(rst),
               .resource_go(resource_go1),
               .rmatrix(rmatrix1));

route_bless2D rc2(.control_in(port2_r),
               .enable(enable_Dim),
               .addrx(addrx),
               .addry(addry),
               .addrz(addrz),
               .addrx_max(max_addrx),
               .addry_max(max_addry),
               .addrz_max(max_addrz),
               .clk(clk),
               .rst(rst),
               .resource_go(resource_go2),
               .rmatrix(rmatrix2));

route_bless2D rc3(.control_in(port3_r),
               .enable(enable_Dim),
               .addrx(addrx),
               .addry(addry),
               .addrz(addrz),
               .addrx_max(max_addrx),
               .addry_max(max_addry),
               .addrz_max(max_addrz),
               .clk(clk),
               .rst(rst),
               .resource_go(resource_go3),
               .rmatrix(rmatrix3));

route_bless2D rc4(.control_in(port4_r),
               .enable(enable_Dim),
                .addrx(addrx),
                .addry(addry),
                .addrz(addrz),
                .addrx_max(max_addrx),
                .addry_max(max_addry),
                .addrz_max(max_addrz),
                .clk(clk),
                .rst(rst),
                .resource_go(),
                .rmatrix(rmatrix4));

age_calf2D age_s1 (.control0_in(port0_r),
                .control1_in(port1_r),
                .control2_in(port2_r),
                .control3_in(port3_r),
                .control4_in(port4_r),
                .control4_ready(port4_ready),
                .clk(clk),
                .control0_out(port0_c1),
                .control1_out(port1_c1),
                .control2_out(port2_c1),
                .control3_out(port3_c1),
                .control4_out(port4_c1));

//always @(*) $display("RC out: %x %x %x %x (rmat %x %x %x %x)", port0_c1,
    port1_c1, port2_c1, port3_c1, rmatrix0, rmatrix1, rmatrix2, rmatrix3);


assign all_valid = port0_r[`calf_valid_f] &
                   port1_r[`calf_valid_f] &
                   port2_r[`calf_valid_f] &
                   port3_r[`calf_valid_f];

    assign validin = port4_cin[`calf_valid_f] & port4_cin[`calf_valid_f];



   assign port4_ready = ( ~(all_valid) |
                       resource_go0 |
                       resource_go1 |
                       resource_go2 |
                       resource_go3)& port4_r[`calf_valid_f]& port_ready_d;



data_buf2D data_s0(.data0_in(port0_di),
                .data1_in(port1_di),
                .data2_in(port2_di),
                .data3_in(port3_di),
                .data4_in(port4_di),
                .clk(clk),
               .data0_out(port0_d1),
               .data1_out(port1_d1),
               .data2_out(port2_d1),
               .data3_out(port3_d1),
               .data4_out(port4_d1));


/******** Stage 2 ********/
wire `routecfg_2w route_config_unbuf;
arbitor2D arb(.rmatrix0(rmatrix0),
          .rmatrix1(rmatrix1),
          .rmatrix2(rmatrix2),
          .rmatrix3(rmatrix3),
          .rmatrix4(rmatrix4),
          .control0_in(port0_c1),
          .control1_in(port1_c1),
          .control2_in(port2_c1),
          .control3_in(port3_c1),
          .control4_in(port4_c1),
          .clk(clk),
          .route_config_unbuf(route_config_unbuf),
          .route_config(route_config));

ctl_xt2D cx(
   .control0_in(port0_c1),
   .control1_in(port1_c1),
   .control2_in(port2_c1),
   .control3_in(port3_c1),
   .control4_in(port4_c1),
   .route_config(route_config_unbuf),
   .clk(clk),
   .control0_out(port0_co),
   .control1_out(port1_co),
   .control2_out(port2_co),
   .control3_out(port3_co),
   .control4_out(port4_co));


data_buf2D data_s1(.data0_in(port0_d1),
               .data1_in(port1_d1),
               .data2_in(port2_d1),
               .data3_in(port3_d1),
               .data4_in(port4_d1),
               .clk(clk),
               .data0_out(port0_d2),
               .data1_out(port1_d2),
               .data2_out(port2_d2),
               .data3_out(port3_d2),
               .data4_out(port4_d2));
   /*********** Stage 3 **********/
   crossbar2D xbar(.data0_in(port0_d2),
               .data1_in(port1_d2),
               .data2_in(port2_d2),
               .data3_in(port3_d2),
               .data4_in(port4_d2),
               .route_config(route_config_unbuf),
               .clk(clk),
               .data0_out(port0_do),
               .data1_out(port1_do),
               .data2_out(port2_do),
               .data3_out(port3_do),
               .data4_out(port4_do));

  always@(posedge clk)begin
  if(!rst)
    port_ready_d <= port4_ready? 0:1;
  end

endmodule
```

الشبكة. و يلاحظ أن قيم المعاملات المذكورة لا يمكن ان تكون عامة و ثابتة لكافة اجهزة التوجيه بدون مخازن و لكنها تعتمد على بنية جهاز التوجيه. أخيرا تم اقتراح أداة تصميم بمساعدة الكمبيوتر مرنة لتقييم الشبكات الثلاثية الابعاد التى لا تحتوى على مخازن. هذة الاداة هى وسيلة بسيطة لاجراء المزيد من التجارب على الشبكات ثلاثية الابعاد بدون مخازن فهي تتيح للمستخدم ضبط و اختيار حجم الشبكة للطبقة الواحدة و اختيار عدد اجهزة التوجيه ثلاثية الابعاد و مواقعها. يحدد المستخدم أيضا جهاز التوجيه و معدل ارسال البيانات. تسمح الاداة بتنفيز الشبكة كما أنه من الممكن ارسال مدونة الشبكة الي المستخدم كمرفق برسالة البريد الالكترونى.

الملخص

مع الزيادة فى عدد المعالجات المتعددة على الرقائق الي العشرات و المئات. اصبح لشبكة الوصل بين المعالجات تاثير كبير علي سرعة اداء الرقائق و الطاقة المستهلكة و التكلفة. بما ان الطاقة المستهلكة فى اطار التواصل بين المعالجات تؤثر فى اداء الانظمة متعددة النواة. لذلك فقد تم طرح اسلوب توجيه جديد يعتمد علي صرف او تحويل مسار المعلومات كبديل أفضل من حيث التكلفة.

اولا يقدم هذا العمل مراجعة علي ادبيات البنية التقليدية لجهاز التوجيه ثم مراجعة علي أادبيات التصميمات التي لا تحتوى علي ذاكرة بعد ذلك تم مناقشة خوارزميات التوجه الخالية من المخزن المؤقت. المساهمة الرئيسئة لهذة الاطروحة هي تطوير جهاز التوجيه الذى لا يحتوى على ذاكرة او مخزن. تقدم هذة الاطروحة جهاز توجيه هو نسخة معدلة من .CONNECTهذا الراوتر الجديد ذو أداء فعال و يشغل مساحة صغيرة. هو يحقق أداء افضل من أجهزة التوجيه المتاحة التى لا تحتوى على مخزن كما انه يشغل مساحة أقل منهم. يشغل Modified CONNECTمساحة اقل من CONNECT ب % 30 و يحقق أداء متقارب منه. غالبا ما تحقق الشبكات التي تتكون من اجهزة توجيه بدون مخازن أداء جيد عند انخفاض حركة مرور البيانات و على أداء منخفض عند ارتفاع حركة مرور البيانات و يزداد الامر سوءا عندما يزيد قطر الشبكة. لذلك تستخدم البنية الثلاثية لتعزز الاداء و لهذا فقد تم تطوير Modified CONNECT ليستخدم في الشبكات الثلاثية الابعاد. تستخدم النسخة ثلاثية الابعاد من جهاز التوجيه المقترح نفس خوارزمية التوجيه المستخدمة في Modified CONNECT و يستهلك جهاز التوجيه طاقة اقل من تلك المستهلكة فى انظمة التوجيه ثلاثية الابعاد المتاحة كما انه يقدم أداء أفضل منهم.

يتضمن هذا العمل دراسة موجزة يتم فيها تقييم عوامل متغيرة في الشبكة من اجل الحصول على القيم الانسب لحجم الشبكات ثلاثية الابعاد في كل طبقة و مواقع اجهزة التوجيه. للحصول على افضل قيم يتم محاكاة خمس شبكات لخمسة اجهزة توجيه مختلفة. معيار الجودة المستخدم في هذة الدراسة يساوى الانتاجية بالنسبة للطاقة الديناميكية المستهلكة في الشبكة و المساحة المشغولة بالشبكة و متوسط زمن انتقال البيانات فى

مهندس:	مروة محمد زكى شاهين
تاريخ الميلاد:	1992/7/21
الجنسية:	مصرية
تاريخ التسجيل:	2014/10/1
تاريخ المنح:	2020/-/-
القسم:	هندسة الالكترونيات و الاتصالات الكهربية
الدرجة:	ماجستير العلوم
المشرفون:	أ. د. محمد رياض الغنيمي
	أ. د. حسام علي حسن فهمى
	د. حسن مصطفى حسن مصطفى

الممتحنون:	أ. د. محمد رياض الغنيمي	(المشرف الرئيسى)	
	أ. د. حسام علي حسن فهمى	(المشرف)	
	د. حسن مصطفى حسن مصطفى	(المشرف)	
	أ. د. أحمد حسين محمد خليل	(الممتحن الداخلى)	
	أ. د. السيد مصطفى سعد	(الممتحن الخارجى)	أستاذ بجامعة حلوان

عنوان الرسالة:

تصميم أجهزة توجيه بدون مخازن للشبكات ثلاثية الابعاد على الرقائق الالكترونية

الكلمات الدالة:

شبكة توصيل المعلومات على رقائق الالكترونية.

ملخص الرسالة:

يقدم هذا العمل مقدمة عن شبكة توصيل المعلومات علي الرقائق الالكترونية ثم مقدمة عن بنيةCONNECT ومقدمة عن تصميمات اجهزة التوجيه بدون مخزن, ثم يقدم هذاالعمل شرح لبنية جهاز التوجيه Modified CONNECT ثم يقدم مناقشة عن مزايا و عيوب الشبكات ثلاثية الابعاد و مناقشة نتائج محاكاة لModified CONNECT ثلاثى الابعاد. يقدم هذا العمل ايضا دراسة تناقش العوامل المثالية للشبكات ثلاثية الابعاد. اخيرا يقدم اداة تتيح للمستخدم ضبط و اختيار عوامل شبكة توصيل المعلومات على الرقائق الالكترونية.

تصميم اجهزة توجيه بدون مخازن للشبكات ثلاثية الابعاد علي الرقائق الالكترونية

اعداد

مروة محمد زكى شاهين

رسالة مقدمة الى كلية الهندسة–جامعة القاهرة

كجزء من متطلبات الحصول على درجة

ماجستير العلوم

فى

هندسة الالكترونيات و الاتصالات الكهربية

يعتمد من لجنة الممتحنين:

الاستاذ الدكتور: محمد رياض الغنيمى	**(المشرف الرئيسى)**
الاستاذ الدكتور: حسام على حسن فهمى	**(المشرف)**
الاستاذ المساعد: حسن مصطفى حسن مصطفى	**(المشرف)**
الاستاذ الدكتور: احمد حسين محمد خليل	**(الممتحن الداخلي)**
الاستاذ الدكتور: السيد مصطفى سعد	**(الممتحن الخارجى)**

استاذ بجامعة حلوان

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية

2020

تصميم اجهزة توجيه بدون مخازن للشبكات ثلاثية الابعاد علي الرقائق الالكترونية

اعداد

مروة محمد زكى شاهين

رسالة مقدمة الى كلية الهندسة–جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
فى
هندسة الالكترونيات و الاتصالات الكهربية

تحت اشراف

أ. د. محمد رياض الغنيمى

استاذ

قسم هندسة الالكترونيات و الاتصالات الكهربية

كلية الهندسة – جامعة القاهرة

أ.د.حسام على حسن فهمى

استاذ

قسم هندسة الالكترونيات و الاتصالات الكهربية

كلية الهندسة – جامعة القاهرة

د. حسن مصطفى حسن مصطفى

استاذ مساعد

قسم هندسة الالكترونيات و الاتصالات الكهربية

كلية الهندسة – جامعة القاهرة

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية

2020

تصميم اجهزة توجيه بدون مخازن للشبكات ثلاثية الابعاد علي الرقائق الالكترونية

اعداد

مروة محمد زكى شاهين

رسالة مقدمة الى كلية الهندسة–جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
فى
هندسة الالكترونيات و الاتصالات الكهربية

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية

2020