A COMPARATIVE STUDY BETWEEN FAT TREE AND MESH NETWORK-ON-CHIP INTERCONNECTION ARCHITECTURES

Azeddien M. Sllame
Computer Network Department
Faculty of Information Technology
University of Tripoli
Aziz239@yahoo.com

Amani Hasan Abdelkader
Computer Science Department
Faculty of Science
University of Tripoli
Tripoli, Libya

KEYWORDS
Network-on-Chip, Fat tree, Mesh, Simulation

ABSTRACT

In this paper we do a comparative study between mesh and fat-tree based Network-on-Chip (NOC) systems. To do the comparison we used two tools; the gpNoCsim and fat-tree simulators. The analysis shows both the efficiency and the applicability of mesh and fat tree structures to NOC targeting system on chip designs. Even though the mesh structure regularity made it suitable for physical implementation, the observed results strongly convinced us to say that the scalability and higher bandwidth of the fat tree structure will make it the preferred interconnection architecture for future massively parallel NOC systems.

INTRODUCTION

System-on-Chip (SOC) is a model that deals with building a system using billions of transistors on a single silicon die (chip). SOC may contain many modules that have variety of signals including digital, analog, mixed-signal, and it often includes radio frequency (RF) functions, buses, memory elements, image processing blocks (e.g. MPEG core), digital signal processing (DSP) cores, general-purpose processors (CPU) and configurable logic blocks (CLB) of field programmable gate arrays (FPGA) and does it all on a single chip as seen in figure 1. Such pre-designed functional blocks (modules, components) are commonly called "cores" intellectual property (IP cores). Nowadays, mobile phones, cable and satellite TV *set-top-boxes*, and portable media devices are typical examples of SOC systems.

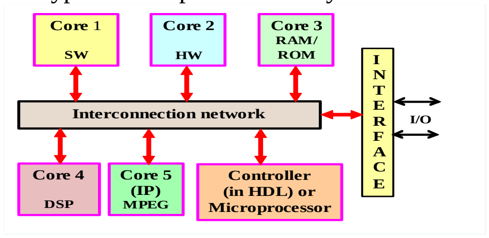

Figure 1: Generic SOC Structure

The most fundamental distinguishing characteristic of a SOC is its structure and connectivity complexity. In practice, most of SOCs are *multiprocessor systems-on-chips* (MPSOCs) because it is too difficult to design a complex SOC without making use of multiple CPUs, various DSP cores, and numerous memory pieces, RF modules. The interconnect topology in the MPSOC model in the last two decades was either point-to-point or bus communication links. Nowadays, instead of connecting the top-level SOC cores by using buses or routing dedicated wires, they are connected to an *interconnection network* that routes packets between them; see both figures 1, 2; which after then named as "*Network-On-Chip*". NOC is a communication subsystem on an integrated circuit (commonly called a "chip"), typically made between IP cores composing of SOC, as seen in figure 2. NOC technology applies networking theory and methods to on-chip communication and brings outstanding improvements over conventional bus and point-to-point interconnections; i.e. "routes packets, not wires" [14].

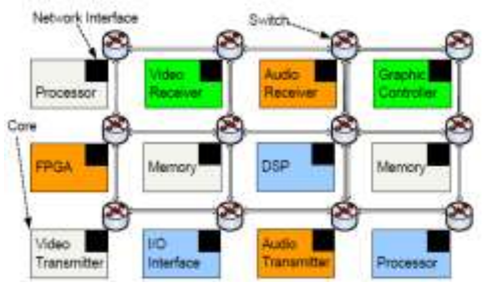

Figure 2: SOC Based on Mesh NOC Interconnection

More precisely, in NOC model, when one IP core is idle, other IP blocks continue to make use of the network resources. Hence, NOC improves the scalability of SOCs, and the power efficiency of complex SOCs compared to other designs. This approach has the benefits of being modular and scalable, well-structured, reusable components, and flexible with higher bandwidth. NOC approach has efficient performance that can be adapted to different workload needs, while maintaining the generality of application development methods and practices. On the other hand, the bus approach has the performance degradation as the number of cores goes beyond 16 cores; and the point-to-point wiring suffers from power dissipation, cross talk delays due to routing inside the chip. Besides that, interconnection networks have been already used in many super-computers for more than five decades, so it is have been systematically researched, thoroughly verified and investigated, and well-documented. Butterfly fat tree (BFT) and mesh architectures are typical examples of those interconnection networks, see figure 2 [1, 2]. Interconnection networks can be classified according to different characteristics. Their topologies fall into two classes *static* (or direct) and *dynamic* (or indirect). In a static network, point-to-point links interconnect the network nodes (IP cores) in some fixed topology; a regular topology as mesh or a hypercube is common examples of this category. A dynamic network allows the interconnection pattern among the network nodes to be varied dynamically: this is accomplished by some form of switching. Examples of dynamic networks include fat trees and multistage interconnection networks. When we speak about networking; we need: *switching*, *routing*, and *flow control* of the packets of the transferred message across that network. The switching technique defines how messages are propagated through the network. More precisely, the switching mechanism defines the hardware and software protocols for transmitting and buffering data when sending a message between neighboring switches (routers) [1], [3]. A variety of switching techniques have been proposed to support communication across the multiple network channels between the source and destination. The most commonly used techniques are; *circuit switching*, *store-and-forward routing*, *virtual cut-through*, *wormhole routing*; more information can be found in [1], [3]. The packet can be defined as the smallest unit of communication containing routing information (e.g. destination address) and sequencing information in its header. Its size is of order of hundreds or thousands of bytes or words. The packet is composed of group of data units named as *flits;* header flit and many data flits. The flit is the smallest unit of information at the link layer; its size is one of several bytes. Flits can be several types and flit exchange protocol typically requires several cycles to transfer one single flit. *Routing* determines the path that will be selected to transfer the packet(s) to reach its destination; it must be decided within each intermediate router which output channel(s) that will be selected to forward incoming packets to their final destinations. *Flow control* defines the synchronization protocol between sender and receiver nodes and determines right procedures to be taken in case of full buffers, busy output channels, faults, deadlocks, etc. In this paper, we are using NOC simulators that allow us to model NOC systems by specifying both the behavior of the network nodes and the way of the communication switching and routing protocols of the whole network.

This paper is organized as follows: interconnection networks are briefly described in section 2. Fat tree network structure is illustrated in section 3. Mesh networks is outlined in section 4. Section 5 outlines theoretical comparison between mesh and fat tree networks. Finally, practical simulation results are given in section 6.

INTERCONNECTION NETWORKS

Interconnection networks have been developed to realize fast and reliable communication systems in parallel systems in last five decades. This fact emphasizes the importance of interconnection networks to overall parallel system performance since any parallel system that employs more than one processor per application program must be designed to allow its processors to communicate efficiently.

FAT TREE ARCHITECTURE CONCEPTS

The fat tree is a type of interconnection network, where the processing elements (IP cores; for simplicity) are interconnected by a tree structure, in which the IP cores are at the leaves of the tree, and the interior nodes are switches. An advantage of a tree structure is that communication distances are short for local communication patterns. Moreover, the fat tree is a tree structure with redundant interconnections on its branches; the number of interconnections increases as the root is reached [9]. The purpose is to increase the bandwidth at higher levels, where it is most needed. Because it is not feasible to provide a channel between every pair of nodes, the network channels are shared among the IP nodes. Messages are used to communicate between sending and receiving nodes, which means construction of paths that are consist some intermediate switches (for switching/routing purposes) along the specified paths from the sources to the destinations. Figure 3 shows a butterfly fat tree with 64 IP blocks (cores) interconnected by suitable number of switches in intermediate levels. The IP nodes are placed at leaves in zero level and switches are placed in higher levels. We can calculate number of levels by the relation:

$L = \log_4 N$, Where *N* is the number of IP nodes.

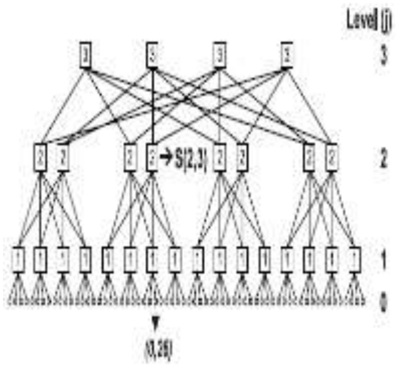

Figure 3: Butterfly Fat Tree Structure with 64 IP Cores

In the network illustrated in figure 3 we have 3 levels, and the switches are placed in levels ranging from $l > 0$ and $l >=$ L. Each IP node is denoted by pair (*i*,0) where *i* is ranging from (0-63) which denotes the index of the IP node in the level zero, each IP node has two ports to connect with its parent switches, each port has two unidirectional physical links. Each level of the network has the number $\frac{1}{2^{(l-1)}} \times \frac{N}{4}$ of the switches and the total number of switches in the network is the summation of the number of switches in each level. Each switch is represented by a pair of coordinates (*i*, *l*), where *i* represents the index of the switch in the level and *l* represents the level of the switch, the pair (5,2) represents switch no. 5 in the level no. 2. Each IP node at the coordinate *(i, 0)* has the parent at coordinate (*p,1*) and *p=i/4*. For example if we have the IP node (0, 62), it has the parent switch (15, 1). Each switch has two parent (*p*) coordinates (*p1,l+1*) and (*p2,l+1*)

$$p1 = \frac{i}{2^{(l+1)}} \times 2^{l} + i \bmod 2^{(2-l)}$$

$$p2 = p1 + 2^{(l-1)}$$

For example, if we have the switch (15,1) , then from the above relations it has the parents:

P1=6 and *p2*=7. And each switch has four children (switches or nodes). The *least common ancestor algorithm* is commonly used with fat tree designs as an adaptive routing algorithm, as seen in [1,10]. In addition, wormhole switching is accompanied with virtual channel mechanism is usually used as a switching technique in recent fat tree designs.

MESH ARCHITECTURE CONCEPTS

Mesh is another type of interconnection networks. 1-D meshes are made from linear arrays of processing elements and incrementally scalable. The most practical meshes are, of course, 2-D and 3-D ones [11]. In a mesh network, the nodes are arranged in a k dimensional lattice of width *w*, giving a total of *wk* nodes.[usually *k*=1 (linear array) or *k*=2 (2D array). Communication is allowed only between neighboring nodes. All interior nodes are connected to *2k* other nodes. The major advantage of the mesh is its simplicity. All links are short and balanced and the overall layout is very regular. The routers are low radix with up to *C+4* input and output ports. The major disadvantage is the large number of hops that flits have to potentially go through to reach their final destination (proportional to "N" for N routers).
XY is a dimension order routing which is a typical minimal turn deterministic algorithm that is commonly used with mesh designs [1,2]. The algorithm determines to what direction packets are routed during every stage of the routing, which routes packets first in x- or horizontal direction to the correct column and then in y- or vertical direction to the receiver. XY routing is working well on both mesh and torus topology. Addresses of the routers are their xy-coordinates, as seen in figure 4.

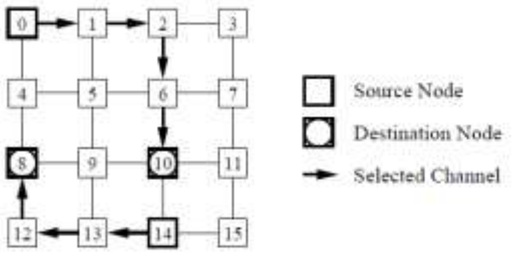

Figure 4: Mesh Interconnection Networks

THEORATICAL COMPARISON BETWEEN FAT TREE AND MESH NOC

In this section we aim to provide detailed comparison between mesh and fat tree as NOC architectures. The data provided here are scattered in many papers with different point of views, e.g. [6], [7], [4], [16], [17], [11], [5]. To the best of our knowledge it is the only study that collected all of such useful information in one document.

Topology:

- Mesh networks belongs to direct type interconnection networks; where point-to-point interconnects the network nodes in fixed regular topology.
- Fat tree is the typical example of the indirect (dynamic) interconnection network; which allows changing of the interconnection arrangement among the network nodes dynamically through the use of network's switches.
- In the 2D mesh topology there are as many switches as IP cores and the design of the switches is always the same (i.e. cost overhead).
- In fat tree topology the number of switches is independent on the number of processing elements and switches with different designs can be used. Packet has many routes to destination due to structure connectivity layout.
- Mesh structure has connections between 4 neighbor nodes
- For mesh topology it is simple to compute the distance between current and destination node as the sum of the offsets in all the dimensions.

Out of order reception of packets :

- Adaptive topology in fat tree allows out-of- order reception of the packets per message flow, which adds complexity to the implementation.
- Deterministic routing in 2D mesh has in-order packet delivery which makes it simple to implement.

Network traffic balance :

- In mesh networks deterministic routing performs well under uniform traffic only.
- In fat tree adaptive routing makes network traffic balance better than deterministic routing thus allowing obtaining higher throughput to the network and low latency.

Deadlock, livelock, starvation:

- Mesh uses XY deterministic routing which is considered as deadlock and livelock free.
- Fat tree designs employ adaptive routing in which there is a possibility of livelock and starvation. Hence, a special care should be taken during switch design process in order to avoid deadlock, livelock and starvation.

Routing:

- Mesh uses XY deterministic routing, in which all packets follow the same route between given source-destination pairs. However, it always chooses the shortest path with in-order flow control.
- In the traditional XY routing, the traffic does not extend regularly over the whole network because the algorithm causes the biggest load in the middle of the network.
- Deterministic routing provides low routing latency and good reliability when the network is not congested.
- Fat tree uses adaptive routing, in which the route can vary from packet to packet depending on the network situation. And it can adapt to network congestion conditions (can do re-routing, out-of-order transmission).

Fault tolerance:

- Mesh structure has low fault tolerance against the possible faults in the network, since it is fixed connected topology.
- Adaptive routing in fat tree increases the fault tolerance. However. Fault tolerance can be achieved via error detection and correction, adaptive routing, but require special attention to avoid deadlocks.

Congestion control:

- Mesh networks can't dynamically respond to network congestion, which will lead to throughput and network efficiency degradation.
- Fat tree can avoid network congestion using adaptive routing, of course by using more information about the network situation. This in turn, increases the complexity of implementation in terms of area and cost.

Latency and throughput:

- In mesh networks the latency and throughput increases with increase in mesh size.
- Adaptive routing is more complex in implementation and provides higher throughput and low latency.

Network utilization:

- XY deterministic routing in mesh networks has under-utilization of the network resources.
- XY deterministic routing tends to send packets toward the center of the mesh when the contention is high. That is in turn will lead to early network saturation and performance degradation.
- XY routing performs well under the uniform distribution traffic.
- Fat tree designs employ adaptive routing in which rerouting and out-of-order enhances network utilization.

Scalability:

- Scalability can be defined as the property which exhibits performance proportional to the number of IP cores employed in the network structure. Moreover, bandwidth (BW) of the network is considered as performance metric.
- Fat tree is more scalable as its BW is the highest as compared to mesh. In addition, fat tree is recursively scalable design.
- The BW in mesh designs is fixed by the number of resources, i.e. it does not scale.
- The BW in fat tree networks depends on the switch design.

Energy dissipation:

- Energy dissipation increases linearly with the increase of the number of virtual channels in switch design for both mesh and fat tree.
- In mesh networks one switch in every network node increases its cost and power consumption.

Physical realization:

- Mesh networks are particularly easily mapped to physical space with uniformly short wires. The simplest case is when the network is a mesh with the same number of dimensions as the physical dimensions of the packaging technology.
- It is not possible to map fat tree topology directly into two dimensions provided by a silicon chip as in the case of mesh topology, without increasing the length of some interconnection wires proportionally to the number of cores. This will decrease the clock frequency dramatically and interference the performance.

PRACTICAL RESULTS

The simulator

General Purpose Simulator for Network-on-Chip Architectures simulator (gpNoCsim) has been described in [12] is used in obtaining practical comparison results. In addition, the fat tree simulator that has been developed under the supervision of the first author, described in [11], is used to validate some results. *gpNoCsim* is an open-source tool developed in Java, component based simulation framework for NOC architectures. Version 1.0 of gpNoCsim contains the implementation of mesh, torus, butterfly fat tree, extended butterfly fat tree networks. Hence, it supports doing comparison of performance parameters among different networks such as throughput, latency (average packet delay), link utilization, buffer utilization, average hop count, average packet not produced. gpNoCsim uses the wormhole switching technique supported with virtual channels in the input and output ports. Figure 5 shows the general structure of the IP cores (nodes) and switch communication flow details. The node defines the necessary methods that are used for messaging and traffic configuration. Messages are generated from a source node and forwarded to its parent switch. From that switch, after appropriate routing and arbitration, the message either forwarded to an adjacent switch or passed to the destination node.

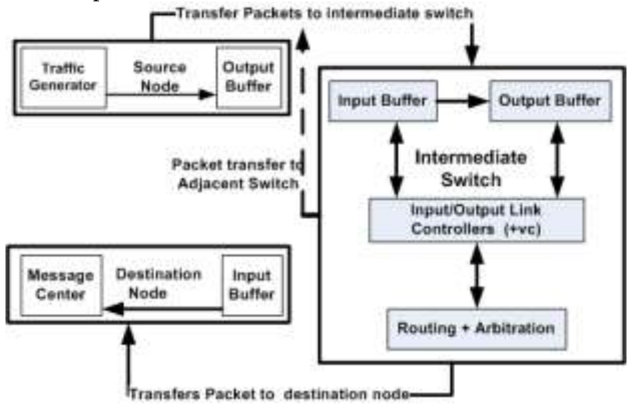

Figure 5: IP Core/Switch Internal Structure and Packet Communication Flow

Nodes are connected to the parent switches through one input physical link and one output physical link. Each node has its address, its generated message list to hold the generated messages, and received message list to hold the received messages from different nodes. Each node has *traffic generator* method which is responsible for packets generation with fixed (or random) lengths, producing random data and sending it to random destinations (it like a seed to feed the network with packets for the simulation be approaching real systems). This method also divides the packet into one header flit and more data flits and one tale flit to indicate the end of the packet [11] , [12].

The analysis

We have analyzed the networks (mesh, fat tree) using the simulator with the following input parameters: IP nodes sizes [16, 32, 64] with different number of virtual Channel / link: [2,4,6,8,10] and with different number of flits / buffer : [2,4,8, 10] with average packet length of [200,300,400] bytes, and number of simulation cycles [1000, 2000, 3000]. Mesh NOC uses XY deterministic routing algorithm, Fat tree uses least common ancestor adaptive routing algorithm. Wormhole switching technique supported with virtual channels mechanism is used in both mesh and fat tree architectures. The following are the main performance parameters in the simulator:

a) *AVG INTER ARRIVAL*: The mean rate of message generation (in simulation cycles) in the resource nodes. Actual inter packet generation interval is calculated by exponential distribution using this parameter. By changing this parameter network load can be increased or decreased.

b) *AVG MESSAGE LENGTH* : The mean length of message (i.e. packet length, in bytes) generated in the IP nodes. By changing this parameter network load can be increased or decreased also.

c) *FLIT LENGTH*: Length of the flit (both header and data flit) in bits. If the size is too small to hold the minimum information to carry header flit then it is increased in number of bits.

d) *VC COUNT* : The number of virtual channels in each of the physical channels in the network, e.g. 2,4,6,8, 10.

e) *FLIT PER BUFFER* : Size of the input buffer and output buffer in flit unit, i.e. fixed amount of buffer is incorporated in input buffer and output buffer of the network.

Table 1 shows some results about the effect of buffering of mesh and fat tree NOC networks when IP cores =16. The results shows a clear increase in average packet delay for mesh as the number of flits/buffer increases, while the increase is not clear in the fat tree. Moreover, in fat tree the number of packets leaving switches is higher than that of mesh.

Table 1: Effect of flits buffering for IP cores =16

Mesh			
Number of flits / Buffer	Throughput	Throughput [Flits leaving Switch]	Avg Packet Delay
2	0.1605	0.54325	45.957446808
4	0.2135	0.781312	65.427350427
8	0.185812	0.66375	56.030927835
fat tree			
2	0.16525	1.1795	43.915254237
4	0.171625	1.185333	43.33620689
8	0.164812	1.120666	50.234234234

Figure 6 shows the results of comparing mesh and fat tree in terms of throughput and number of virtual channels per one physical link. The number of flits/buffer is constant=2. Both designs have nearly linear throughput when virtual channels changes from 2 till 6. But, we see that in the fat tree case the throughput started decreasing when the number of virtual channels reaches 8 per physical channel due to switching overhead, while mesh network drops when virtual channels =8 and increases again. Figure 7 illustrates the relationship between average throughput per switch [Flits leaving Switch] and number of virtual channel for 16 IP cores networks with the number of flits/buffer =2. AS shown in the figure 7, the fat tree has higher throughput per switch which comes from adaptive routing used within the fat tree switch. Figure 8 clarifies the relationship between average packet delays (ns) with average message length (bytes) for 64 IP cores networks. This figure produced when the number of virtual channels =4 for each physical link and number of flits/buffer=4. The figure shows that fat tree has higher average packet delay than mesh networks when the packet length= 300,400 bytes. Figure 9 describes the relation between average packet delay and number of virtual channnels for 64 IP cores. We used here constant number of flits/buffer =2 and the change were in number of virtual channel. As shown in the figure 9 the fat tree has higher average packet delay than mesh and it goes down in a linear fashion as the number of virtual channels increase. But the mesh behaved differently in different values. The relation between number of VC/link and number of flits stored in buffers for 64 nodes is illustrated in figure 10. The figure shows buffer size (2 to 8) flits per buffer for fat tree NOC. For every virtual channel no./link there are 3 situations for buffers. As the number of buffers increase the average packet delay decrease in the network for most of the cases. On the other hand figure 11 illustrates the same parameters for mesh networks. However, as the number of buffers increase the average packet delay has not changed too much for most of the cases for mesh networks.

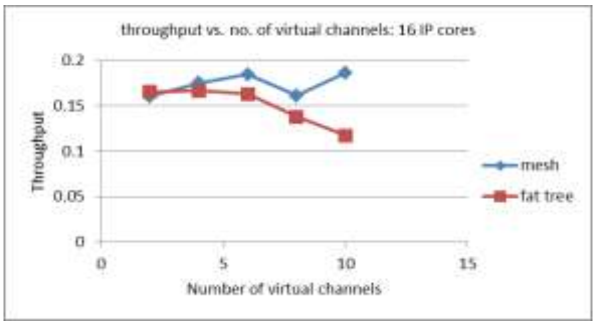

Figure 6: Throughput and Number of Virtual Channels for 16 IP Cores

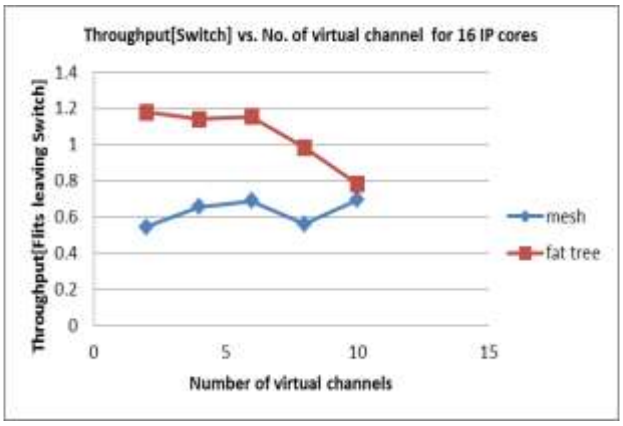

Figure 7: Average Throughput per Switch (Flits Leaving Switch) and Number of Virtual Channel for 16 IP Cores

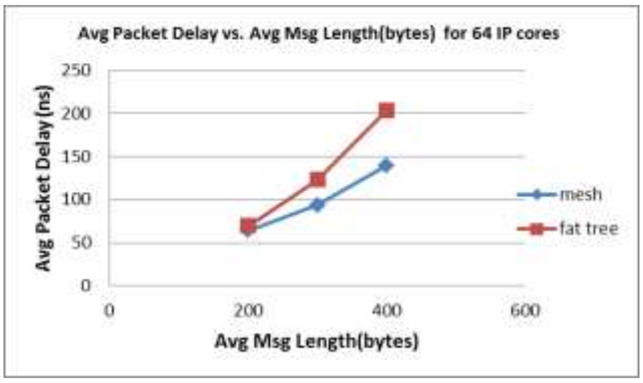

Figure 8: Average Packet Delays (ns) with Average Message Length (bytes) for 64 IP Cores

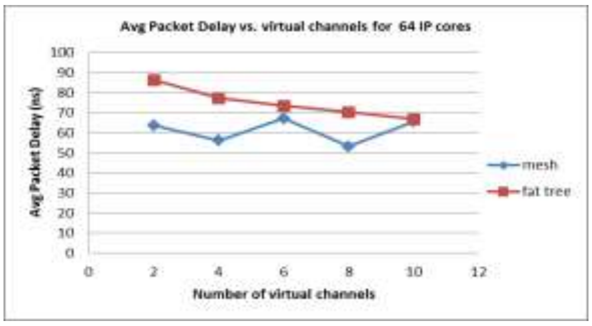

Figure 9: The Relation Between Average Packet Delay and Number of Virtual Channnels for 64 IP Cores

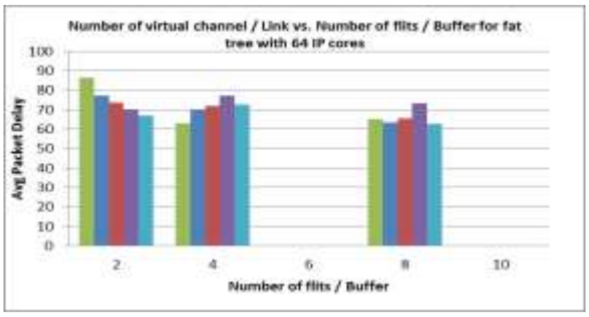

Figure 10:. Different Values of Buffer Size with Different Virtual Channels for Fat Tree with 64 IP Cores

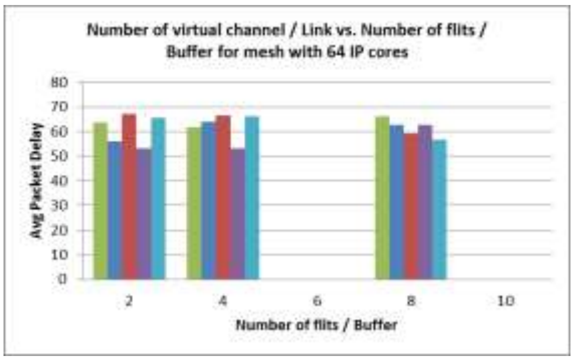

Figure 11: Different Values of Buffer Size with Different Virtual Channels for Mesh with 64 IP Cores

CONCLUSION

The main goal of this paper was to analyze the 2D-mesh and fat-tree architectures as a NOC interconnection networks candidates. The evaluation process has been done using available open-source simulators. The comparison includes: *routing*, *switching methods used in the switches*, *effect of buffering*, *effect of virtual channel technique*, *effect of packet length*. We believe that the scalability and higher bandwidth of the fat tree network makes it the preferred NOC for future massively parallel NOC systems. Even though some papers reported that the most common topology that now mostly used in NOC design is 2D mesh due to structure regularity which helps physical mapping (60% of NOC cases [17]).

REFERENCES

1. Duato J., S.Yalamanchili, L. Ni. 2002. *Interconnection Networks–An Engineering Approach*. Morgan Kaufmann Publishers, San Francisco.
2. Duato, J., O. Lysne, R. Pang, and T. M. Pinkston. 2005. “Part I: A theory for deadlock-free dynamic reconfiguration of interconnection networks.” *IEEE Trans. on Parallel and Distributed Systems* 16, No. 5 (May), 412–427.
3. Dally, W. J., and B. Towles. 2004. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, San Francisco.
4. Dally W. J. and B. Towles. 2001. “Route packets, not wires: On-chip interconnection networks.” In *Design Automation Conference*, pp. 684–689.
5. Pande P. P., C. Grecu, A. Ivanov, R. Saleh. 2003. “Design of a Switch for Network-on-Chip Applications.” In *Proceedings of ISCAS*, Bangkok, Vol. V, pp. 217-220.
6. Kumar S., et al, “A Network on Chip Architecture and Design Methodology.” In *Proceedings of ISVLSI*, pp. 117-124, 2002.
7. Wingard D. 2001. “Micro Network-Based Integration for SoCs.” In *Proceedings of DAC2001*, LasVegas, Nevada, USA.
8. Guerrier P., A. Greiner. 2000.”A generic architecture for on-chip packet switched interconnections.” In *Proceedings of DATE Conference and Exhibition*, pp. 250 –256.
9. Grecu C., P. P. Pande, A. Ivanov, R. Saleh. 2004. “A Scalable Communication-Centric SoC Interconnect Architecture.” In *IEEE International Symposium on Quality Electronic Design ISQED*, San Jose, California, USA, 22-24 (March).
10. Leiserson C. 1985. “Fat-Tree: Universal Network for Hardware-Efficient Supercomputing.” *IEEE Trans. On Computers* C-34, No. 10 (Oct.), 892-901.
11. Sllame A. M., A. Elasar. 2012. “An Efficient Switch for Fat Tree Network-on-Chip Interconnection Architecture” In *IEEE International Conference on Computer Systems and Industrial Informatics (ICCSII'12)*, Sharjah, UAE.
12. Bononi L., N. Concer, M. Grammatikakis, Coppola, M. R. Locatelli. 2007. “NoC Topologies Exploration Based on Mapping and Simulation Models.” In *Proceedings of the 10th Conference on Digital System Design Architectures, Methods and Tools*. IEEE, CA.
13. Hemayet H. M., A. Ahmed, T. Islam AI-ayeemn and A. Md Mostofa. 2007. “GPNOCSIM - A General Purpose Simulator for Netwotk-on-Chip.” In *Proceedings of the International Conference on Information and Communication Technology IEEE ICICT07*.
14. Dale K. P. 2004. “Modeling and simulation verification and validation challenges.” *Johns Hopkins APL Technical Digest* 25, No. 2, 163-172.
15. Al-Tawil K. M., M. Abd-Elbarr, F. Ashraf. 1997. “A survey and comparison of wormhole routing techniques in a mesh networks.” *IEEE Network* 11, 38–45.
16. Salminen E., A. Kulmala, and T. Hamalainen. 2007. “On network-on-chip comparison.” In Euromicro DSD, 503–510 (Aug).
17. Jantsch A, H. Tenhunen. 2003. *Networks on Chip*. Kluwer Academic Publishers, USA.

BIOGRAPHY

Azeddien M. sllame (BSc, MSc, PhD) received his B.Sc in Computer Engineering in 1990 from Engineering Academy, Tajoura, Libya. He earned his M.Sc in Computer Science and Technology from Brno Technical University, Czech Republic in 1997. In 2003 he got his PhD in Information Technology from Brno University of Technology. He published more than 25 scientific papers in many international conferences and journals in the areas of designing of high-performance digital systems, system-on-chip, network-on-chip designs and evolvable hardware systems. He is now assistant professor with faculty of information technology (University of Tripoli, Libya). Email: aziz239@yahoo.com.