Applying MPLS Technique as On-Chip Communication Means for Network-on-Chip with Mesh Topology

Azeddien Sllame, Nagwa Salama
University of Tripoli
Tripoli, Libya
Aziz239@yahoo.com, Nagwa771@gmail.com

Hadeel Youns Ben Rajab
Technical Electronic College, Ben Ashour
Tripoli, Libya
h.younis@bsisp.ly

***Abstract*—This paper describes designing efficient mesh topology Network-on-Chip system by employing MPLS protocol as an on-chip communication technique. Discrete-event simulator is developed in C++ to show the applicability of MPLS in providing efficient on-chip communication for multicore processing system-on-chip designs. Experimental results are compared with two simulators: wormhole equipped with virtual channels; and MPLS-based fat tree network-on-chip systems. Outstandingly, MPLS as on-chip communication means recorded better results than the wormhole +virtual channels in terms of throughput and packet end-to-end delay (latency).**

I. INTRODUCTION

The multicore processors are becoming the key success of multiprocessing and power-efficient computing. Current embedded applications such as real-time multimedia applications necessitate intensive computation and higher communication's bandwidths which are difficult to handle by different kinds of bus structure such as multi-bus or hierarchal-bus as on-chip communication means on traditional SoC systems. Consequently, powerful on-chip communication architecture is required to support the full functionality of these applications. Besides that, multicore principle enforced designers to shift implementing embedded systems from system-on-chip (SoC) into multiprocessing SoC (MPSoC), whereas heavy data transfer between such multicores lead designers to use interconnection networks as on-chip communication technique between heterogeneous multicores composing of MPSoC system, as shown in Fig. 1 [1], [2], [3]. However, the use of interconnection networks imposed applying the principle of separation between computation and communication inside the MPSoC systems, as it has been previously realized in well-known parallel computing and supercomputing machines and systems. Therefore, cores inside the chip implementing MPSoC system start communicating by sending packets through interconnection network instead of using old fashioned electrical buses which use wires routed around the chip [4]. Thus, interconnection networks allowed the current MPSoC systems to be implemented as networks inside a chip; which is named as network-on-chip (NoC); for more information about interconnection networks see [5], [6]. NoC systems are a scalable networks built inside a chip based on using interconnection networks for on-chip communication; that capable of interconnecting a large number of heterogeneous cores. Scalability, energy efficiency, and reliability are the main important advantages of this new on-chip communication model as borrowed from previously well-researched interconnection networks. In addition, NoC structure provides fault tolerance feature by using adaptive routing protocols which can use different paths for communication between cores. Furthermore, NoC design paradigm increases engineering productivity in such a way that NoC eliminates ad-hoc global wire engineering, as well as it has advantages of modular, well-structured, flexible structures.

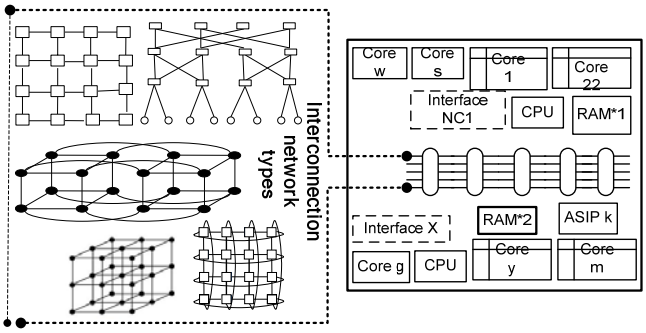


Fig. 1. MPSoC system with different types of interconnection networks

However, NoC is a complete platform for system design, heterogeneous cores interfacing, hierarchical integration, routing and flow control, debugging and testing [2]. Therefore, recent improvements in MPSoC design techniques and advances in deep submicron chip technologies opened-up the feasibility of a wide range of applications making use of massive parallel processing and tightly interdependent processes, some adhering to real-time requirements, bringing into focus new complex aspects of the underlying communication structure, which facilitated by NoC efficiently. With NoC implementation an aggregate bandwidth grows while the bandwidth of the bus is shared and the speed goes down as the cores added to the bus implementation. In addition, NoC system designs are based on the interconnection networks which exhibit pipelining concurrency in their designs as they have been developed to work with parallel computing and supercomputers. Whereas, bus implementations did not have any concurrency, pipelining is difficult to implement with buses. Furthermore, the NoC designs demonstrate the separation of abstraction layers. Bus designs suffer from computation-communication separation feature. However, unlike NoC designs, the bus designs are fairly simple to build

with low cost, while NoC designs need new design techniques and methodologies, such as network interfacing and novel router (switch) designs [3],[7],[8]. This paper presents the applicability of using MultiProtocol Label Switching (MPLS) networking protocol suite as an on-chip communication protocol within mesh interconnection network for future NoC designs. From networking point view MPLS is an intelligent forwarding mechanism which integrates efficient switching functions on layer 2 (data link layer) with effective routing of layer 3 (network layer) technique to provide high-performance scalable networks to allow efficient use of network resources such as routers; with the capability of applying different quality-of-service (QoS) levels according to various applications' requirements [9], [10], [11].

This paper demonstrates how we can exploit the capabilities of the MPLS in terms of efficient label switching, path creation between communicating cores and routing speed over label-switched paths on the NoC design paradigm. However, MPLS inserts labels to packets, specify QoS level, establishes paths among communicating cores based on labels which maps link-specific labels in the packet header to outgoing links of the router (switch), then forwards packets over connected paths; where packet-forwarding decisions at the routers are completed quickly based only on the contents of the inserted label, without the need to examine the whole IP address of that packet as found in TCP/IP networks. Consequently, such rich features of MPLS and the work in [11] inspired us to develop a C++ simulator to research on the applicability of MPLS to realize efficient mesh NoC systems.

This paper is organized as follows: Previous work is reported in section 2. Mesh interconnection network are briefly described in section 3. MPLS technique's principle is given in section 4. The realized mesh NoC system is described in section 5. Section 6 illustrates the simulator structure. Section 7 reports the performance analysis results. Concluding remarks are described in section 8.

II. II. Previous Work

There are many related research projects arranged in many universities and research centers which are supported by different international research organizations such as IEEE. Current state-of-the-art in multicore processing SoC systems focus on NoC design tools, routing algorithms, power management and design methodologies by different aspects with diverse point views targeting different interconnection network topologies such as fat tree, 3D mesh, torus, and 2D mesh; see [1] and [3]. Some representative of such related researches are: Tobias and Shanker in [3] presented a detailed survey of research and practices of NoC systems design in which they showed the relation between cores, networking interfaces and switching/routing components inside the NoC architecture. In addition they discussed the switching methods and QoS practices, with no mention to MPLS technique. Nikolay and Gerard in [8] provided a comprehensive survey about on-chip communication systems which talked over routing, queuing, flow control with a discussion of mesh topology. Li-Shiuan Peh et al in [12] presented some NoC implementations based on mesh topology. Brett and Partha in [13] evaluated mesh and tree-based NoCs and reported that they both are accomplished better performance when implemented in a 3D chips more than the 2D realizations. However, they claimed that the mesh-based NoC systems demonstrated improved performance in terms of throughput, latency, and energy dissipation with an extra area added as a chip space overhead. In [14] L. Bononi et al offered a comparison analysis of ring, 2D-mesh, and Spidergon using theoretical uniform traffic using request/reply on a MPEG4 benchmark example. D. Ludovici, et al. in [15] described a research work about physical synthesis of Fat trees and 2D mesh topologies. They provided that as the size of the NoC system scales up, 2D meshes will suffer from poor performance scalability, whereas Fat trees will gain better performance scalability with area cost. Basavaraj et al in [16] presented a simulation tool written in System-C that is able to search for NoC designs with mesh and torus topologies for HW implementation using label switching protocol (LDP) which is one of MPLS assisted protocols. R. Kurdy et al in [17] showed a research work which equates the performance between IP and MPLS with fat tree NoC system with recovery mechanism using high level simulation in NS-2. However, authors have not shown any design details; just described upper-level simulation study using NS-2 which is used in modeling and simulation of traditional networks. Azeddien Sllame and Asma Elasar in [18] testified a Fat tree based NoC system with a wormhole routing + virtual channels with no use of MPLS mechanism in their described system. Azeddien Sllame and Nagwa Salama in [19] described a Fat tree based NoC system with MPLS technique and provided a comparison against wormhole routing + virtual channels which showed improvements to MPLS in terms of throughput and latency.

On the other hand, the work in this paper is a continuation of research work described in [19]. The discussion here is focused on applying MPLS technique on 2D mesh topology to produce efficient mesh NoC system. Therefore, the design of 2D mesh NoC system will include all necessary components, procedures, and functions for switching, arbitration, routing and buffering with MPLS networking mechanism. In addition, a comparison analysis with the work in [20] which describes NoC systems with wormhole routing + virtual channels will be provided.

III. Mesh Interconnection Network

Interconnection network is a communication structure connects a group of processing nodes by means of a specified number of communication switches to form network architecture in a particular topology. Switches are responsible for switching, routing, and flow control of packets flowing over the interconnection architecture between the processing cores. Network topology specifies the structure details such as the static organization of nodes and number of stages of the network and switches. Interconnection networks can be categorized according to different characteristics. Their topologies fall into two main classes *static* (or direct) and *dynamic* (or indirect). In a static network, point-to-point links interconnect the network nodes in some fixed topology; a regular topology as mesh or a hypercube is common examples of direct type. A dynamic network permits the interconnection arrangement between the network nodes to be changed dynamically which is made by employing some switching/ routing techniques; such as in Fat trees and multistage interconnection networks [5], [6].

Nowadays, 2D mesh structure is the most used NoC topology because of its simple structure which maps very well in VLSI physical realization. Linear arrays are called 1D mesh, while the 2D mesh structure is the practical topology that are frequently used; see Fig. 2; although the 3D mesh structure are sometimes used but not as the 2D mesh. In a mesh topology network, the processing cores are organized in a *k* dimensional lattice of width *w*, giving a total of *wk* processing cores. (When k=2 we get the 2D mesh layout). In mesh the communication is allowed between neighboring cores which are directly interconnected by switches and their switches are interconnected by communication links. Simply speaking, mesh topology is regular, simple architecture, in which each internal switch is connected to 4 neighbor routers (switches), with short links. Each router is connecting only to one processing core either with bidirectional or separate in-out links, each with sufficient buffering. However, XY routing which uses a typical minimal turn deterministic algorithm is commonly used with 2D mesh topologies, where the algorithm determines to what direction packets are routed during every stage of the routing by routing packets in row (X: horizontal direction) firstly then it turns to route the packet to the correct column (Y: vertical direction) toward targeted core. Though, the XY routing is working well with mesh topology, since addressing is described by the routers' positions that are represented by XY coordinates. To be more precise the deterministic minimal routing means the selected path is one of shortest paths between the sender and receiver and is totally determined only by sending and destination cores, thus every taken hop toward the destination core makes the packet closer to the receiving core [5] [6].

Features of the mesh topology [5] [6] [8]:

- ***Topology:*** mesh network has a fixed regular topology that belongs to direct class of interconnection networks. Meshes arranged as matrixes with switches at intersection points of rows; with each switch has point-to-point links with its neighboring switches; processing core is attached also to its parent switch with a direct link too. Therefore, as the XY-dimensions of mesh increases the number of switches (routers) increases, which adds more cost overheads, which disturbs the scalability.
- ***Network traffic balance:*** in 2D mesh topology the deterministic routing incurs in-order packet forwarding over the same path which produces simple designs which efficiently working under uniform traffic only.
- ***Deadlock, livelock, starvation:*** mesh uses XY deterministic routing that is thought-out as deadlock and livelock free.
- ***Routing:*** mesh uses XY deterministic routing; in which traffic does not spread regularly over the entire network because the algorithm causes the major load to flow in the middle of the mesh. Consequently, in mesh topology the deterministic routing produces low routing latency and good reliability when the network is not under congestion status.
- ***Fault tolerance:*** mesh as an interconnection network has low fault tolerance capability since it is fixed connected topology.
- ***Congestion control:*** due to the fixed regular organization of mesh topology; it is difficult to respond dynamically to congestion when happened. This in turn has a side effect on network efficiency and throughput.
- ***Latency and throughput:*** In mesh networks, as the size of the network increases the latency and throughput will increase, but this requires to take care about heavy traffic congestion.
- ***Network utilization:*** with mesh topology the XY deterministic routing may causes the network resources to be underutilized under non-uniform traffic, since the traffic flow in the center of the mesh is more than outside of the mesh; which will overwhelm the capacity of the links of the switches in the center of the mesh. This may increase reaching the congestion state very quickly and the mesh performance will be disturbed. Alternatively, under uniform traffic distribution the XY routing working well.
- ***Scalability:*** performance normally increases with the increase of processing cores which will efficiently utilize the bandwidth of the network. However, due to the mesh topology inherited structure as a regular fixed organization it is not scalable as the number of processing cores increases.
- ***Energy dissipation:*** the energy dissipation increases linearly as the number of processing cores, switches, and links increase during the operation of the network.

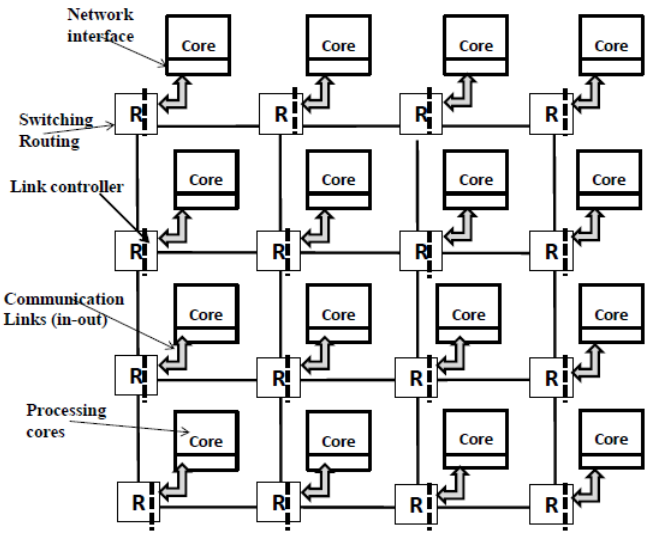


Fig. 2. Mesh topology (4X4 matrix)

IV. MPLS PRINCIPLES

MPLS is a communication protocol standardized by Internet Engineering Task Force (IETF) that developed to accommodate communication with other many protocols; which enables MPLS to establish and maintain a connection-oriented network over connectionless networks such as TCP/IP network or makes easy label encapsulation in other network types such as making packet-over-SONET/SDH, or label over ATM PVCs, label Frame Relay PVCs, and Ethernet [9]. In TCP/IP networks the packet decision for forwarding it from router to the next hop router along the paths is made at each router by inspecting the whole IP header of that packet; which consumes processing power of each router and incurs some delays to the packet transmission. Unlike TCP/IP networks, the MPLS shortens the delay and reduces the processing power consumption of the packet forwarding process along the path from sending core to receiving core [9]. However, MPLS attaches short label to the packet at the entrance router of the

MPLS domain with proper QoS service level; which will be used by subsequent routers to forward the packet along specified path. Thus, MPLS routers made the packet forwarding decision only by inspecting that short label which will reduce the transmission delays and processing overhead of the packet switching and forwarding processes inside the router; along the path. Simply, within the MPLS network; the label is used as an index into a label-table that identifies out-interface of current router directly, and then the old label is replaced with a new label of the in-interface of the next-hop router. Hence the packet is forwarded to its next hop very rapidly [9] [10]. Another powerful feature of MPLS protocol is that the label contains a field named as forwarding equivalence class (FEC), which enables classification of packet streams, where all packets that associated to a specific FEC and which transferred from an identified node will follow the same path or a set of certain paths, and got the same treatment by all routers along the path. However, in the MPLS domain, routers are using label distribution protocol (LDP) to create labelled switched paths (LSP) by performing label to a FEC mapping, where each MPLS routers maintains learned labels in label forwarding information base (LFIB); each entry of LFIB associates an FEC with an (LDP Identifier, label) pair, to maintain LSP path as a series of interconnected (in-to-out) labels. Thus, when next hop changes for a FEC, MPLS router retrieves the label for the new next hop from the LFIB. However, LFIB is extracted from general label information base (LIB) table; hence LFIB is smaller than the ordinary routing forwarding table found in TCP/IP networking; which gives the forwarding speed to MPLS over TCP/IP networks. MPLS operation details found in [9] [10]. Finally, because of lower packet transfer delay, efficient path establishment and resource utilization, rapid forwarding technique, scalability, and assured performance of the services offered by MPLS mechanism; MPLS is seen as one of the most suitable networking technologies for fulfilling transmission requirements for real-time voice and video applications, as reported in [21][22]. Those MPLS features encouraged us to apply MPLS as on-chip communication technique into NoC system, since many current MPSoC systems such as those found in mobile computing devices include working with real-time applications beside different other Internet applications.

V. The Realized Mesh NoC System

The implemented mesh NoC system contains three major components: switch, processing core, and mesh network. In this section these three parts are briefly described.

A. Switch structure

Switching technique describes the HW and SW protocols that are needed for transmitting and buffering data during message sending between adjacent switches (or routers) of any network. The designed switch is based on MPLS switching/ routing technique, with bidirectional links where every port is linked with a pair of opposite unidirectional channels, for transmitting and receiving. However, all switches are designed as the same LER MPLS routers that capable of performing ordinary IP routing (TCP/IP) and MPLS switching with MPLS forwarding methods, see Fig. 3.

The switch contains the following components.

(1) Crossbar switch: It is the central module of the switch which is in charge of connecting switch input buffers to switch output buffers. Crossbar offers more speed to further enhance MPLS with more speed. However, by limiting the crossbar's size on in-out the scalability will not be affected.

(2) Link controllers: link controllers are input link controllers and output link controllers that interconnect switches and define the network topology. In this switch design every link controller at input ports and output ports have links to the three packet types and buffers (IP, MPLS, and LDP). Buffers sores different packets temporarily, which scales the bandwidth as the number of ports increases and requires a switch with a number of input ports equal to the number of physical channels. Thus, the switch has a buffer management unit to observe buffers of different packets (IP, MPLS, and LDP).

(3) Routing unit: MPLS mechanism is considered as a table-based routing technique because of building many tables (routing table, routing forwarding table, LIB, and LFIB). However, both the Hamiltonian paths and XY routing algorithm are used to support the MPLS operation to help making of such tables [5] [6].

(4) Arbitration unit: MPLS protocol is a control protocol so the LSP path has to be established between communicating cores before sending any packet, which is sufficient to use only a simple arbitration technique to control the switch shared resource: switch buffers and switch in/out ports. However, a simple fair arbiter is employed to treat all links and requests at each cycle equally.

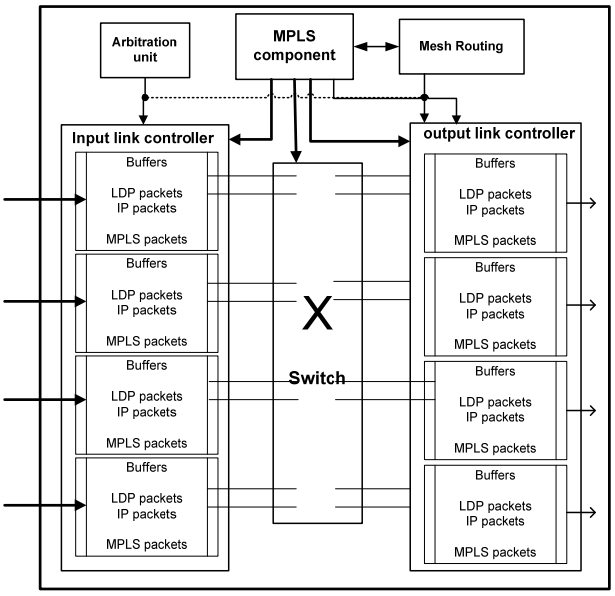


Fig. 3. Switch internal structure

(5) MPLS component: responsible for applying all functions and procedure of MPLS technique; which include:

- **LSPs creation**: this procedure constructs LSP path from the source switch to the destination switch, where the first ingress LER switch controls and assigns FEC by using source/destination IP addresses. A new LSP path is created at LIB table and a request label message (LDP packet) is sent to the neighboring switch to perform the mapping of the label and create a new LSP in LFIB table at every switch the

packet will pass until the packet reaches the parent switch of the destination processing core. After that the LDP packet will return back through the same path carrying a mapping label to complete the desired path on LFIB. Finally the LSP is activated in LFIB table at ingress switch; the state diagram is illustrated in Fig. 4.

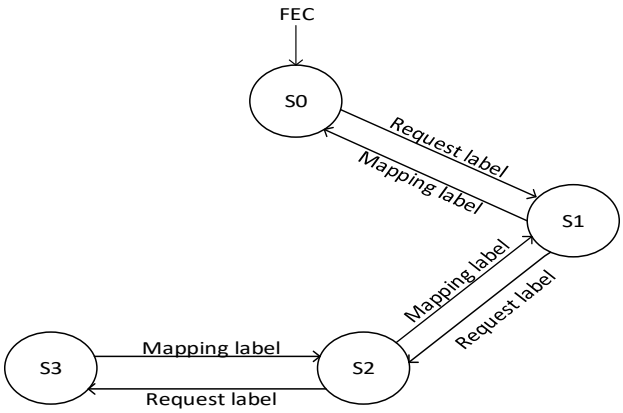


Fig. 4. LSP's creation process using LDP protocol of MPLS

As shown in Fig. 4 the *S0* is representing the ingress MPLS LER switch that defines the FEC of the packet, as well as it starts the initiiation of LSP path connecting source with destination using the LDP protocol associated with MPLS. Hence, the *S0* will add a new entry to its FIB table *S0* will define the output interface to *S1* inside the FIB table. Then *S0* sends a request label to S1, when S1 responds with the mapping label, a new entry will be added to the LFIB table where it has the output interface assigned with the label and the input interface with the assigned label, this process keeps repeating at each switch until it arrive at the egress LER switch. Therefore, this will return the mapping to S2 until it reaches S0 and then LFIB entry will be completed and the desired LSP path is finished.

(6) Forwarding unit of packets using MPLS technique: Forwarding of packets through a particular mesh NoC system is made simple by applying MPLS technique over the routers (switches) enclosed in the selected LSP path. Therefore, the forwarding of packets is continued until the packets reach the destination switch by consulting the LFIB table and determining the outgoing label and the definite interface of each switch. However, as the packet reaches the target switch, the switch will strip out the label and forwards it as traditional IP packet to its final processing core. The state diagram of this task is depicted in Fig. 5. As seen in Fig. 5; state S0 receives an IP packet from processing node N0. As mentioned each switch in the proposed mesh NoC network works as an edge LER MPLS switch. Thus, S0 will assign FEC to the incoming IP packet and defines a label to it, then; it sends it as MPLS packet. Accordingly, S1 receives that MPLS packet and based on the LFIB table along the LSP; it searches it and will swap the incoming label with the next output label and sends it through the specified outgoing interface as an MPLS packet. Therefore, the same routine will be repeated in S2 and transfers the MPLS packet to S3, where also the S3 will pull the label off as the LER edge switch (as a last switch) and passes the traditional IP packet to the final destination processing node N1.

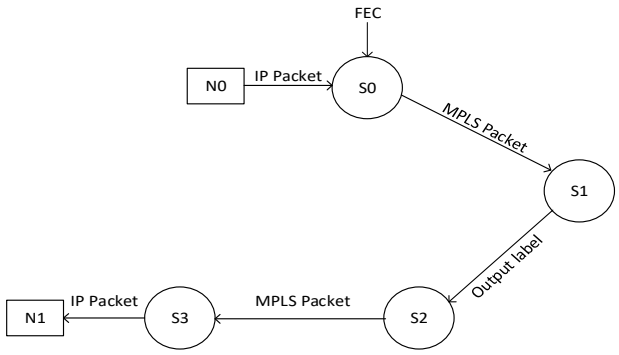


Fig. 5. State diagram of packets forwarding using MPLS technique

(7) Other switch functions: the switch contains four main algorithms with their related functions and procedures, which perform the switching of different packets (IP, MPLS, LDP) across the switches include:

- Moving LDP packet from the switch's input buffer to the switch's output buffer
- Moving LDP-packet from switch's output buffer to the neighbor switch input buffer
- Moving MPLS-packet from switch's input buffer to switch's output buffer
- Moving MPLS-packet to the neighboring switch's input buffer

B. Processing node (core) internal structure and functions

The processing node is designed as a separate unit that models and simulates the processing core of mesh topology of NoC system which simply works by generating messages (packets), sends packets to other cores, buffer packets temporarily, and consumes the received messages (packets) from other cores inside mesh network. The internal structure that clarifies the operation of a processing node as a source or as a destination with its relation to the parent switch is illustrated in Fig. 6.

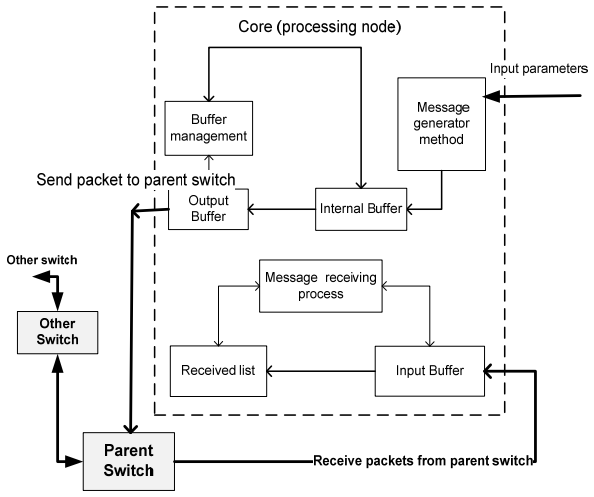


Fig. 6. The simulator communication flow model

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 produced messages, and received message list to hold the received messages from different nodes. The node component in the developed C++ program contains whole needed methods which used for messaging and traffic configuration. Messages are generated from a source node and forwarded to its parent switch. From that switch, with the aid of MPLS switching and forwarding components and arbitration, the message either transferred to an adjacent switch or supplied to the destination processing core.

C. Mesh network generation process

The network is the main structure at the highest abstraction level of the modeled mesh NoC system. The mesh NoC network generation piece of software implements the operations of generating, transferring, and receiving of the messages between all processing nodes of the mesh NoC in the form of packets. The network entity achieves the function of generating the mesh network structure by creating a list of cores (nodes) and a list of communicating MPLS switches, setting adjacent switches which make the whole mesh topology network, and mapping of nodes with switches, as seen in Fig. 2. Also, this entity implements the transferring function which forwards packets through the network from output buffer of a node to the parent switch's input buffer. Furthermore, this entity performs the forwarding of packets from switches output buffers to the nodes input buffers. Basically, it is called upon request from the switch object and performs the method of switching/moving packets from node's input buffer to node's output buffer across the crossbar. **The mesh network generation component realizes the following:**

(i) Generation the mesh network method

The network object in the developed mesh NoC design creates all the processing cores, performs nodes linking between every node with its parent switch, and then interconnects all the switches to make the required mesh structure NoC network; the steps of this method cover:

1) Firstly calculate the number of switches by the relation: *Number of switches=number of IP nodes.*
2) Determine the position of each switch as row or column (matrix);
3) Create the switch;
4) Add the switch to the switch list;
5) Create a processing node and link it to its parent switch;
6) Add addressing to every switch;

(ii) Setting adjacent switches

This method completes the generation of the mesh network by setting the adjacency relationship between the switches in the mesh matrix layout; defining row and column, e.g. illustrating example shows (4X4 matrix).This method accomplishes the following steps:

1) Define the position of each switch;
2) Compare the row of the switch to the max. number of rows;
3) If the row is smaller than the max. number of row then the position of the adjacent switch will be in the top;
4) If the row is equal to the max. number of rows then the position of the adjacent switch is in the same row;
5) If the row is bigger than 0 then position of the adjacent switch will be in the bottom, and if it equals 0 then it's in the same row;
6) Compare the column to the max. number of columns;
7) If the column is smaller than the max. number of columns then the position of the adjacent switch will be in the right, if its equal then it is in the same column;
8) If the column is bigger than 0 then position of the adjacent switch will be in the left, and if it equals 0 then it's in the same row;
9) Go to the next switch in the switch list.

VI. SIMULATOR STRUCTURE

The target architecture of the designed NoC system is the mesh topology which highlights simplicity, simple routing mechanism and somehow network scalability. However, the mesh switches are arranged in dimensional matrix in a columns and rows arrangement in which the processing nodes (cores) are interconnected to the switches. Remember that the node has a message generation unit that makes messages in a random way, transform every message into a series of packets depending on the packet's length that is defined in the simulator setup file, and store those packets inside the node internal buffers. In addition, the core owns output buffer unit in order to briefly store packets before transferring them toward the adjacent switch. Furthermore, the core contains input buffer element to receive the incoming packets sent to it from other processing nodes (cores) across the parent switch. The simulator running procedure is shown in Fig. 7.

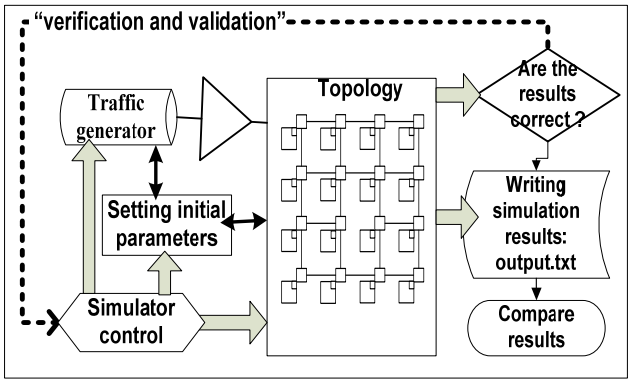


Fig. 7. Simulator running procedure

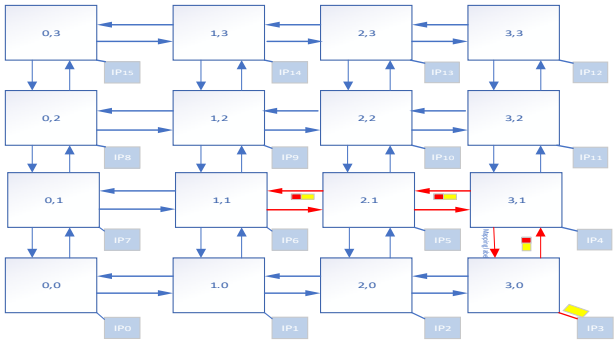
The packet transfer between communicating nodes (cores) is made through the LSP paths using different switches of the mesh topology. However, the on-chip communication process running inside the developed NoC system is implemented by applying MPLS technique with associated flow control and packet forwarding mechanism which uses label assignments to IP-packets using accurate LSP construction procedure that based on the mappings of (FEC, Labels) to LSP paths, which is the function of LDP protocol. The communication flow is quite complex, since every node generates random messages, every message contains specific number of packets, then packets are temporarily stored in node's output buffers to be ready for sending to a parent switch's input buffer whenever there are free buffer cells available at the parent switch's input buffer. Subsequently, those packets become ready to be routed and passed from switch-to-switch by label insertion/deletion

(swapping) through the defined LSP path until the packet reaches the final switch. However, the final switch then passes the packet directly to its final destination node's input buffer for final processing inside the node (core).

The simulator has the following features:

- The simulator shows the MPLS label advertising and label bindings with FECs that are made possible by running LDP protocol to create LSP paths exploiting the matrix of switches found in mesh topology before transmitting any data packets between any cores inside the NoC system, as seen in Fig. 8.
- The simulator shows the movement of label request process, label and FEC mapping, label insertion to packets, illustrates the MPLS packets after creation, label swapping during packet switching, switching of MPLS packets across switches along the selected LSP path, and demonstrates label stripping when packet arriving at its destination.
- The simulator generates a random number of messages for each node; each message has a random length. Then, each message is divided into a series of packets. These packets are sent out to random nodes.
- The simulator shows the number of generated messages from each processing core (IP node), for each message the simulator also presents while running the total number of its constructed packets.
- The simulator demonstrates the movement of packets (forwarding of packet) sent over switches along different paths inside the network. Also, it shows all sent/received packets by all cores at every simulation cycle.
- For each packet, the simulator marks on cycle-by-cycle basis the LSP path taken by that packet from sender to destination.
- The simulator provides an output log file that records all the simulation work on packet's level.
- The simulator at the end of simulation process reports the average packet latency (end-to-end delay), number of successfully received packets, and number of lost packets, calculate core's throughput, and estimate the system throughput.

The developed simulator has been tested with 4x4 mesh topology NoC system shown in Fig. 8 which has 16 cores (intellectual propriety nodes (IP nodes)), Fig. 8-a illustrates simulator initial configuration is listed in Fig. 8-b. Fig. 8-c shows simulation cycle #1 and cycle #2446 as a portion of the simulator's output file, which is clear evidence that the simulator works with efficient MPLS functions and procedures. In addition it describes how the simulator will use the previously explained theories and designed components such as how LDP creates LSP paths, crossbar and switching, MPLS forwarding and routing procedures.



(a) Mesh topology as 4X4 matrix NoC example

```
MPLS Mechanism [Packets]....
Mesh Network Configuration....
---------------------------------------
Number of IP nodes-->16
Fixed Message Length --> false
Avg Msg Length (bytes)--> 6656
Packet Lenght(bytes)128
Avg Msg Production Rate--> 250
Number of physical link/Switch --> 5
Number of Packet/Buffer 4
Number of Cycles-->100000
Press any key to continue
```

(b) Simulator's initial configuration

```
-------------------
|      Cycle # 1
-------------------
Moving traffic from IP node output buffer to parent Switch
26 Packets is transferring From Node[4] To Node[1] Cycle 1
GenTime 1
• 20 Packets is transferring From Node[11] To Node[6] Cycle
  1 GenTime 1
• 1 Packets is transferring From Node[12] To Node[5] Cycle 1
  GenTime 1
--------------------------
Arriving packets to switches
• Packet  is arriving To Switch (3,1) from Node[4]
• Packet  is arriving To Switch (1,2) from Node[9]
• Packet  is arriving To Switch (3,2) from Node[11]
• Packet  is arriving To Switch (3,3) from Node[12]
--------------------------
Moving LDP Messages between Switches.....
--------------------------
•Request Label Message of FEC 401 is transferring From
 Switch (3,1) To Switch (0,1)
•Request Label Message of FEC 1106 is transferring From
 Switch (3,2) To Switch (3,3)
•Request Label Message of FEC 1205 is transferring From
 Switch (3,3) To Switch (3,0)
•Request Label Message of FEC 906 is transferring From
 Switch (1,3) To Switch (1,2)
•Request Label Message of FEC 1205 is switching From Switch
 (3,0) To Switch (3,3)
•Request Label Message of FEC 401 is switching From Switch
 (0,1) To Switch (0,2)
•Request Label Message of FEC 906 is switching From Switch
 (1,2) To Switch (1,3)
•Request Label Message of FEC 1106 is switching From Switch
 (3,3) To Switch (3,2)
--------------------------
|      Cycle # 2446
--------------------------
Moving traffic from IP node output buffer to parent Switch -
 --------------------------
16 Packets is transferring From Node[7] To Node[10] Cycle
 2446 GenTime 2446
```

```
• Packet  is arriving To Switch (0,1) from Node[7]
• Packet  is arriving To Switch (2,3) from Node[13]
-------------------------
Moving LDP Messages between Switches...........
-------------------------
•Request Label Message of FEC 710 is transferring From
 Switch (0,1) To Switch (0,2)
•Request Label Message of FEC 710 is switching From Switch
 (0,2) To Switch (1,2)
-------------------------
Moving MPLS Packets between Switches...........
-------------------------
•MPLS Packet of Tag 65 is Swapping its Tag by Tag 73 in
 Switch (2,0 )
•MPLS Packet of Tag 73 is Popping its Tag in Switch (3,0)
 and Forward to its IP destination # 3
•MPLS Packet of Tag 125 is Swapping its Tag by Tag 65 in
 Switch (2,1 )
•MPLS Packet of Tag 207 is Swapping its Tag by Tag 125 in
 Switch (2,2 )
•Tag 207 is pushing to Packet # 34 in Switch (2,3)
•MPLS Packet of Tag 73 is transferring From Switch (2,0) To
 Switch (3,0)
•MPLS Packet of Tag 65 is transferring From Switch (2,1) To
 Switch (2,0)
•MPLS Packet of Tag 125 is transferring From Switch (2,2) To
 Switch (2,1)
•MPLS Packet of Tag 207 is transferring From Switch (2,3) To
 Switch (2,2)
-------------------------
Arrive IP Packets to its Destination:
-------------------------
Node[3] has received Packet 30 from Node[13]
Final statement of the simulator's out put
 - Throughput[Net] -----> 1.57837
 - Throughput[Packets Leaving Switch]-> 5.76989
 - Node Throughput -----> 1.57856
 - Averag Packet Delay ----->26.3531
 - Total Generated Messages: 6203
 - Total generated packets: 157,875
 - Total Sent Messages: 6202 (157,842 packets)
 - Total Recieved Messages: 6200(157,835 packets)
```

(c) Portion of the simulator's output

Fig. 8. Some parts of the output log file of the simulator

VII. Performance Analysis Results

At the end, the achieved end results of the developed simulator are compared with the results that are obtained from two other simulators the MPLS FAT tree NoC simulator which described in [19], and the gpNoCSim simulator that has been described in [20]. The gbNoCSim simulator is an open source NoC simulator based on wormhole routing enhanced with virtual channel mechanism as an on-chip communication means for mesh NoC system.

A. Experiment (1): Running simulators

The first experiment is to compare 16 node topologies: (4X4 mesh) and 16 node Fat tree. The results are shown in tables 1, 2, and 3. However, Table I shows of 16 cores Fat tree topology results using MPLS reports that the average packet delay of 21.73ms with 26368 generated packets with a throughput of 0.26 with zero lost packets, this result is the best result among the all reported results because of redundancy links of Fat tree structure. Table II illustrates the results of the gbNoCSim simulator with wormhole+virtual channels technique. The maximum generated messages 42642 recorded the average packet delay of 111.04 ms with 9 lost packets which is 5 times more than that of MPLS Fat tree with a throughput value equals to 0.164. Table III describes the results MPLS network simulator results for 4x4 mesh NoC system which reported average packet delay of 38.58ms with 115732 generated packets with a throughput of 0.921 with 8 lost packets. Therefore, MPLS with both topologies (mesh and Fat tree) reported lowest average packet delay than wormhole+virtual channels technique. But, the mesh topology with MPLS reported the worse results of average packet delay than Fat tree topology because of Fat tree includes redundancy links.

TABLE I. MPLS network simulator's results for Fat tree 16 nodes NoC system

Generated packets	Sent packets	received packets	Lost packets	MPLS Throughput	Avg Packet Delay
58222	58222	58222	0	0.58	22.72
41017	41017	41017	0	0.40	21.77
26368	26368	26368	0	0.26	21.73
17069	17069	17069	0	0.16	21.12
11477	11477	11477	0	0.11	20.82
9563	9563	9563	0	0.08	20.71
2392	2392	2392	0	0.02	21.87

TABLE II. gbNoCSim simulator results for 4x4 mesh NoC System

generated packets	Sent packets	receivd packets	Lost packets	wormhole Throughput	wormhole Avg Packet Delay
42642	42633	42633	9	0.164	111.04
25629	25629	25629	0	0.098	79.50
21275	21275	21273	2	0.082	78.80
12787	12787	12786	1	0.048	68.73
9884	9884	9879	5	0.039	71.99
8465	8465	8463	2	0.032	68.52
6408	6408	6406	2	0.024	68.41

TABLE III. MPLS network simulator results for 4x4 mesh NoC system

Generated packets	Sent packets	received packets	Lost packets	MPLS Throughput	Avg Packet Delay
101731	101731	101726	7	1.647	44.06
115732	115732	115724	8	0.921	38.58
107128	107128	107120	8	0.80	36.50
80441	80441	80440	1	0.671	39.02
57138	57138	57132	6	0.49	36.50
30009	30009	30007	2	0.41	37.75
25790	25790	25780	10	0.21	38.29

B. Experiment (2):Troughput comparisions

In this experiment a comparison between the throughputs of the developed simulator for MPLS-based mesh NoC with MPLS-based Fat tree NoC systems against the gbNoCSim simulator which uses wormhole+ virtual channel mechanism.

However, the node throughput is measured by counting the packets that sent by the source node and safely arriving at the destination node over a specified time interval. Whereas, the total throughput of the whole network is measured by accumulating all delivered traffic flows; i.e. (total received packets between all processing cores multiplied by packet's length) dividing it by (number of processing cores multiplied by total cycle of the simulation [20]. Fig. 9 illustrates the throughput comparison of 8X8 mesh with MPLS against 8X8 mesh with wormhole+virtual channel mechanism of gbNoCSim simulator, the figure clearly demonstrates that MPLS technique gained about 5 times better throughput than the wormhole+virtual channel mechanism. However, with MPLS the throughput is linearly increasing as the load of injected packets increases. Another kind of comparison results is shown in Fig. 10, which shows throughput with MPLS technique for 64 cores Fat tree and mesh topologies. The figure remarks that mesh with MPLS technique and with the increase of processing cores (64 cores) the mesh still gains better throughput than Fat tree with MPLS. The recorded results approves also that the mesh throughput is still increases linearly with the traffic increase and overcomes the fat tree for the size of 16 and 64.

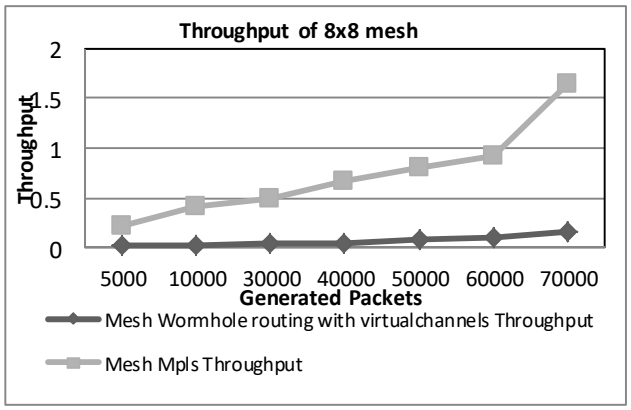


Fig. 9. Throughput of 8X8 mesh with MPLS against 8X8 mesh with wormhole+virtual channel mechanism

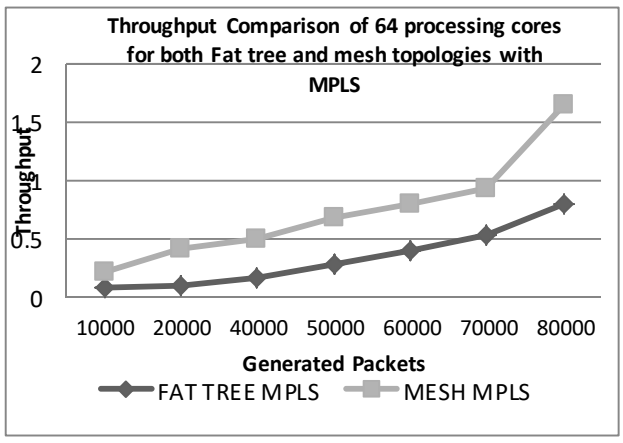


Fig. 10. Throughput with MPLS technique for 64 cores Fat tree against mesh

C. *Experiment (3): Average packet delay*

In this experiment an assessment of average packet delay between the developed simulator with MPLS-based mesh NoC system against the MPLS-based Fat tree NoC system with a contrast of gbNoCSim simulator which uses wormhole+ virtual channel mechanism. However, the latency can be used as a performance evalution parameter for interconnection network. Latency (end-to-end delay) can be defined as the time needed to complete a transfer of a packet from a source core to a destination core. Therefore, the average packet end-to-end delay for a network topology can be calculated by dividing the summation of all the recieved packets' delays by all procesing cores over the number of all received packets by all cores to get the value of latency in cycles/packet [20]. Fig. 11 describes the average packet delay for 8x8 mesh (64 cores) by comparing MPLS against wormhole+virtual channels techniques, which show that there is a big variation between the wormhole +virtual channels when compared to MPLS in 8x8 mesh topologies. However, the figure results clearly showing that MPLS has gained nearly constant end-to-end packet delay value even with increase of the traffic load, while wormhole+virtual channels has reported 3-times higher end-to-end packet delay which increases also as the traffic starts to cross 25000 packets/sec, which indicates the start of network congestion. In addition, the average packet delay with MPLS technique is less than the one registered with wormhole +virtual channels with a value factor of at least 1.5. Another analysis result is shown in Fig. 12 for the average packet latency (end-to-end delay) which worked out between Fat tree and mesh topologies using MPLS in both same size topologies; with 64 cores. The figure clarifies that the Fat tree topology has slightly less average packet delay than mesh structure, that due to the redundancy of the communication links of Fat tree that is added to the topology to make it with more links as we go up toward the root. Therefore, Fat tree topology is better than mesh topology in terms of average packet end-to-end delay.

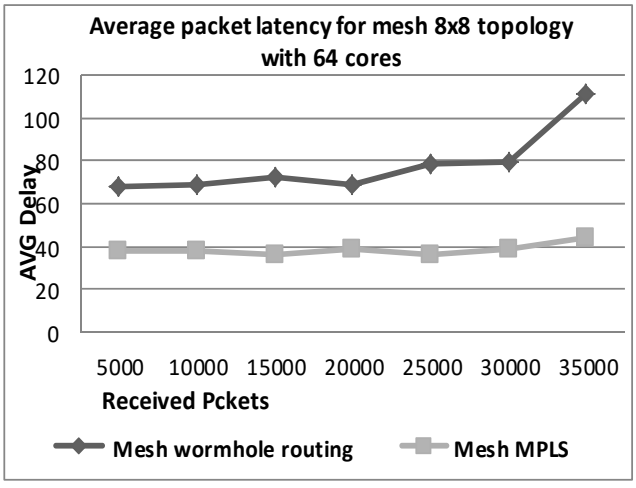


Fig. 11. Average packet delay for mesh topology with MPLS against wornhole+virtual channels

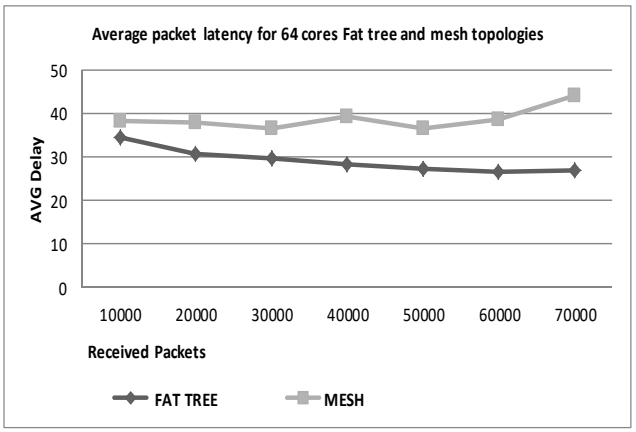


Fig. 12. Average packet delay for 64 cores with MPLS for Fat tree and mesh topologies

VIII. CONCLUSIONS

MPLS is described as an efficient on-chip communication technique for mesh NoC design to realize MPSoC systems. A simulator written in C++ is built to verify all the proposed functions and procedures to make MPLS as on-chip communication means such as input/output link controllers, crossbar switch, MPLS switching and mesh routing units including LDP protocol for LSP path establishment, buffer management. In addition processing IP cores are designed with sufficient methods required to packetize, send, store, transfer, receive, and forward messages. The presented results of network throughput and average packet latency (end-to-end packet delay) demonstrate that the MPLS technique is suitable to implement efficient mesh NoC system when equated by wormhole switching+virtual channels. Finally, comparison analysis with Fat tree topology found that the Fat tree overcomes the mesh topology in terms of average packet latency because of the available redundancy links found in upper levels of fat tree structure toward the root that facilitates more routes and provides scalability to fat tree topology.

REFERENCES

[1] A. A. Jerraya and W. Wolf (editors), *Multiprocessor Systems-On-Chips*. San Francisco, USA: Morgan Kaufmann Publishers, 2005.

[2] A. H. Jantsch and H. Tenhunen, *Networks on Chip*. USA: Kluwer Academic Publishers, 2003.

[3] Tobias Bjerregaard, and Shankar Mahadevan: "A Survey of Research and Practices of Network-on-Chip", ACM Computing Surveys, Vol. 38, Article no.1, March 2006.

[4] Dally, W. J., and B. Towles, "Route packets, not wires: On-chip interconnection networks", Proc. of the DAC'38 Conference, Las Vegas, June 2001.

[5] J. Duato, S. Yalamanchili, L. Ni, *Interconnection Networks – An Engineering Approach*, USA: Morgan Kaufmann, 2002.

[6] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, San Francisco: Morgan Kaufmann Publishers, 2004.

[7] De Michelli G, Benini L. *Network on Chips*, Berlin: Morgan Kaufmann: 2006.

[8] Nikolay Kavaldjiev and Gerard J. M. Smit: "A survey of on-chip communications for SoC,

[9] Rosen E., Viswanathan A., and R. Callon: "Multiprotocol Label Switching architecture", IETF RFC 3031, 2001.

[10] M. Deepankar and R. Karthikeyan, *Network Routing, Protocols, and Architectures*. USA: Morgan Kaufmann Publishers, 2007.

[11] Dominguez-Dorado M., Rodriguez-Perez F. J., Gonzalez-Sanchez J. L., Marzo J. L., Gazo A., 2005: "An Architecture to provide Guarantee of Service (GoS) to MPLS", IV Workshop in G/MPLS Networks.

[12] Li-Shiuan Peh, Stephen W. Keckler, and Sriram Vangal: On-Chip Networks for Multicore Systems, Chapter 2 in S.W. Keckler et al. (eds.), Multicore Processors and Systems, Integrated Circuits and Systems, DOI 10.1007/978-1-4419-0263-4_2, Springer Science+Business Media, 2009.

[13] Brett Stanley Feero, Partha Pratim Pande: Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation, IEEE Transactions on Computers, Vol. 58, No. 1, January 2009, pp.32-45.

[14] Luciano Bononi, Nicola Concer, Miltos Grammatikakis: NoC Topologies Exploration based on Mapping and Simulation Models, IEEE, August 2007.

[15] D. Ludovici, et al., "Assessing Fat-Tree Topologies for Regular Network-on-Chip Design under Nanoscale Technology Constraints", Proc. of Conf. on Design, Automation and Test in Europe, 2009.

[16] Basavaraj Talwar, Shailesh Kulkarni and Bharadwaj Amrutur, "Latency, Power and Performance Trade-offs in Network-on-Chips by Link Microarchitecture Exploration", 22nd Intl. Conference on VLSI Design, Jan. 2009.

[17] Reza Kourdy, Mohammad Reza Nouri Rad, Mohammad Pooyan, Majid Rahimi Nasab, "Improvement MPLS-NOC bandwidth by dividing bandwidth in Fat-tree topology", In the 2nd International Conference on Computer and Automation Engineering (ICCAE), Singapore, IEEE conference, Feb. 2010, Volume: 3, pp. 470 – 474.

[18] Azeddien M. Sllame and Asma Elasar: Modeling and Simulating Network-on-Chip Designs: A Case Study of Fat Tree Interconnection Architecture, In International Journal of Computer Theory and Engineering, pp. 823-829, Vol. 5, No. 5, October 2013.

[19] Nagwa Salama and Azeddien M. Sllame: Designing an Efficient MPLS-Based Switch for FAT Tree Network-on-Chip Systems, In ACM Proceedings of the 1st International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems (AISTECS workshop), Prague, Czech Republic, January 2016.

[20] 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.

[21] Azeddien M. Sllame, Mohamed Aljafry: "Performance Evaluation of Multimedia over IP/MPLS Networks", International Journal of Computer Theory and Engineering, Vol. 7, No.4,pp.283-291, August 2015.

[22] J. Evans and C. Filsfils, *Deploying IP and MPLS QOS for Multiservice Networks: Theory and Practice*. USA: Morgan Kaufmann Publishers, 2007.