International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 2, No. 1, March 2013, pp. 27~48
ISSN: 2089-4864

Configurable Router Design for Dynamically Reconfigurable Systems based on the SoCWire NoC

Arash Farhadi Beldachi*, Mohammad Hosseinabady, Jose L. Nunez-Yanez***
* Department of Electrical and Electronic Engineering, University of Bristol
** Department of Electrical and Electronic Engineering, Queen's University

Article Info

Article history:

Received Dec 1, 2012
Revised Jan 22, 2013
Accepted Feb 6, 2013

Keyword:

SoCWire NoC
SoCWire Router
Partial reconfiguration
Configurable router
Evaluation platform

ABSTRACT

New Field Programmable Gate Arrays (FPGAs) are capable of implementing complete multi-core System-on-Chip (SoC) with the possibility of modifying the hardware configuration at run-time with partial dynamic reconfiguration. The usage of a soft reconfigurable Network-on-Chip (NoC) to connect these cores is investigated in this paper. We have used a standard switch developed with the objective of supporting dynamically reconfigurable FPGAs as the starting point to create a novel configurable router. The configurable router uses distributed routing suitable for regular topologies and can vary the number of local ports and communication ports to build multi dimensional networks (i.e., 2D and 3D) with different topologies. The evaluation results show that the selection of the ideal router is different depending on traffic patterns and design objectives. Overall, the mesh network with a four local ports router offers a higher level of performance with lower complexity compared to the traditional mesh with one local port router.

Corresponding Author:

Arash Farhadi Beldachi,
Department of Electrical and Electronics Engineering,
University of Bristol,
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK.
Email: arash.beldachi@bristol.ac.uk

1. INTRODUCTION

State of the art FPGAs have large densities capable of implementing complete multi-core System-on-Chip (SoC) with the possibility of modifying the hardware configuration at run-time with partial dynamic reconfiguration [1]. The communication needs of these multicore designs create *performance bottlenecks* and *power*, *scalability* and *reliability* issues [2].

Network-on-Chips (NoC) have been proposed as an alternative to bus based designs capable of overcoming these bottlenecks [3]], [[4]. A NoC offers much higher flexibility compared with the traditional point-to-point communication thanks to the usage of routers which are key components in a NoC [5]], [[6]. Utilizing a NoC in an FPGA-based reconfigurable platform to connect multi-cores is a challenge mainly due to the lack of a technology independent and easy implementable communication infrastructure capable of supporting partial reconfiguration.

In this work, the concepts of NoC and dynamic reconfiguration are merged and the possible synergies investigated. To do this, we have employed SoCWire [7]], [[8] which is a NoC design based on the ESA SpaceWire standard [9] to design a configurable router named SoCWire Router. We have selected the SoCWire Switch to create SoCWire Router due to Hot-plug ability to support dynamic reconfigureability, link error detection and recovery ability in hardware and robustness. In addition, it is technology-independent, supports different FPGA vendors and its packet level flexibility allows the implementation of a wide range of communication protocols.

In this paper, switch means a switch which has been designed for a source path addressing network and a router is a switch which has been built to support algorithmic routing. The currently available SoCWire can be used to support various network topologies but with significant addressing overheads for regular networks. It uses a source path addressing scheme which is efficient for small networks. We have modified the SoCWire Switch to create a low overhead router called SoCWire Router for regular networks with many nodes adding logical addressing. In addition, the proposed router can be configured with varying number of local ports for 2D and 3D networks which make it flexible to use in the different regular topologies. The proposed modified router (i.e., SoCWire Router) keeps the ESA SpaceWire standard and SoCWire features.

We have created different configurations of the SoCWire Router to design a number of equivalent networks with a fixed number of computing nodes. To verify the capabilities of the networks, we have designed an on-chip performance evaluation platform, examined the partial reconfiguration ability by using the equivalent networks as the dynamic and rest of the performance evaluation platform as the static part and investigated the equivalent networks performance with realistic and randomized traffic patterns in a physical prototype.

The rest of the paper is organised as follows: Section 2 reviews previous research, briefly introduces the SoCWire NoC in the background sequel and motivation for the extension of the SoCWire Switch to SoCWire Router is presented in this section. Section 3 introduces the proposed configurable router and investigates different SoCWire Router configurations and the configuration we have used in this paper. In section 4 a number of equivalent networks based on the different configurations are built. In addition, our on-chip performance evaluation platform which is designed to measure the equivalent networks performance under a synthetic and realistic traffic loads is introduced in this section. Section 5 investigates the effect of adding different buffer sizes to the SoCWire communication ports. The equivalent networks are evaluated in section 6 with the partial configuration technique.Finally, Section 7 concludes the paper.

2. LITERATURE REVIEW, BACKGROUND AND MOTIVATION

This section reviews previous research, explains briefly the SoCWire NoC architecture and presents the extension motivation of the SoCWire Switch to SoCWire Router.

2.1. Previous Research

There have been several proposed reconfigurable SoC with different communication infrastructures and we are going to discuss some of them.

ReCoBus [10] introduced a technique to generate On-Chip buses suitable for dynamic partially reconfigurable platforms.

The Artemis NoC [11] is an infrastructure that supports specific reconfiguration services which explores the Hermes NoC [12] infrastructure by adding a set of services. One of the services is adding an interface between reconfigurable IP and router which is two macros to remove the possible produced glitches effects in the interface between the under reconfiguration IP and the rest of the device and may produce false data into the Network which causes malfunctions or circuit blocking. The second service is discarding the transmitted packets to an area which is under reconfiguration because it is not clear the packets are targeted to the previous or next configuration of that area.

Dynamic Network-On-Chip (DyNoC) [13]], [[14] proposed to provide the end user with a valid framework to support flexible inter-cores communication. The purpose of the DyNoC architecture is to make the cores reachable at any instant of time and from any module within the system via switching elements surrounding the cores. The dynamic capabilities of DyNoC are due to its support to dynamic module placement. The DyNoC consist of a two dimensional mesh interconnected routers that have a defined position in the reconfigurable system, and therefore they can be overwritten during the reconfiguration process if more room is needed for loading a new core. The overwritten routers are deactivated and packets are routed through the routers that are surrounding the newly loaded module, using SXY which is a modified XY routing algorithm to dynamically bypass the possible obstacles. The DyNoC architecture and the surrounding routing algorithm have been validated with a light controller control system in a 3x3 DyNoC mapped on a Virtex II FPGA. In contrast the SoCWire Router use the regular XY routing algorithm for two dimensional meshes, easily supports different regular topologies and in this work the NoC based on the different configuration of the SoCWire Router is mapped as a single block to a centralized area of the device with processing elements connected in the periphery. This approach is effective at using the current design flows of partial reconfiguration since it is possible to use the available FPGA resources optimally. If the number of local/communication ports for multidimensional networks change in a router the wiring infrastructure will change as well and this can be achieved by letting the P&R FPGA tools manage the resources in the assigned communication area without imposing excessive constraints.

Communication Unit Network (CuNoC) [15] presented as an extension to DyNoC. This work does not use any buffering mechanism at the input ports and uses a priority to the right rule in the concurrent incoming scenario. The same as DyNoC provides the possibility of introducing a new module, mapping it to one or more tiles of the mesh and the same drawback as the DyNoC in comparison to our work.

Configurable Network-On-Chip (CoNoChi) [16] provides a flexible and dynamically reconfigurable infrastructure for dynamically reconfiguration SoCs. This work is based on the idea that switches can be also removed and or added to the network like the modules. CoNochi consists of switches with four bidirectional links. These links can be used for connections to adjacent switches or for connecting a hardware module to a switch. CoNochi makes it possible to dynamically insert/remove the modules and switches to/from the network. The main drawback is that the length between switches might dynamically change and cause link delay. Also, the defined grid FPGA distribution implies that all the unused areas of the grid have to be configured to connect at least default communication lines.

Reference [17] proposed a run-time reconfigurable NoC framework. This NoC framework can dynamically create/delete express lines between SoC components (implementing dynamically circuit-switching channels) and perform run-time NoC topology and routing-table reconfigurations to handle interconnection congestion, with a very limited performance overhead. This works reveals the employment of a set of specific NoCs instead of a single static NoC and makes it possible to improve the performance and consumed power. The communication infrastructure in this work does not provide the flexibility and scalability which can be provided by a SoCWire Router based NoC.

The work at [18] proposed a multiplayer approach to interconnect modules. The switch multiplexers are reconfigured to connect different modules by employing the fine grain partial reconfiguration and change the network topology via interconnecting available network signals. A mix bus-ring communication has been mapped in the architecture by switching four bit multiplexers implemented with four input LUTs. There is no more information available regarding further implementation and this communication infrastructure is not scalable for a many cores SoC.

Modarressi et al. [19] presented a reconfigurable architecture for NoC on which arbitrary application-specific topologies can be implemented. When a new application starts, the inter-router connections in the NoC change to some predefined pattern depends on the application traffic pattern and changes the topology. This NoC architecture requires many dynamic configuration switches which increases by the size of the network.

There are some academic and commercial designs based on the SpaceWire standard such as SpaceWire Light [20], SpaceWire (SpW) [21], SpaceWire UK [22], 4links drivers products [23], Atmel AT7910E [24], SoCWire and GRSPWROUTER [25]. SoCWire, SpaceWire (SpW), and SpaceWire Light are open source and available online. The key point is that all of the works based on the SpaceWire standard have designed a routing switch which is not suitable for regular networks because of the lack of algorithmic routing. To the best of our knowledge, this is first work:

a) To propose a configurable router based on the SpaceWire standard for the SoCWire NoC which supports partial reconfiguration systems and deployed it in the construction of different mesh topologies with varying configurations.
b) To employee a configurable router which supports partially reconfiguration systems, develop a performance evaluation platform and investigate NoC performance with realistic and randomized traffic patterns in a physical prototype.

2.2. Background

As SoCWire has been used and modified in this paper, in the sequel, we explain briefly the overview of the SoCWire NoC architecture [8]. The ESA SpaceWire standard is a well-established standard and a proven interface implementing a serial link bi-directional asynchronous communication protocol. The standard includes features that make it especially suitable for Space applications such as link initialization and reconnection following a disconnection, error detection and recovery and hot-plug capability. The SoCWire is a NoC design based on the SpaceWire standard has been developed by IDA, Technical University Braunschweig, with the objective of supporting dynamically reconfigurable FPGAs in future Space applications which is based on the standard features of SpaceWire while increasing its throughput by using parallel interfaces. The SoCWire verification results have shown Partial Reconfigurable Modules (PRMs) in the applied test cases do not have an effect on the host system and SoCWire is suitable for dynamic partial reconfiguration applications [26].

The SoCWire uses the following control characters for flow control: Flow Control Token (FCT), End of Packet (EOP), and Error End of Packet (EEP) and Escape Character (ESC) form the higher level control code. FCT indicates that there is space for 8 more normal characters in the receiver buffer and EOP clarifies the end of a packet. EEP determines the packet is terminated prematurely due to a link error. ESC is

used to form the high level control code which is 8-bit in length like NULL and Time-Code. NULL (ESC + FCT) keeps a link active, and Time-Code (ESC + data character) distributes system time information over the SpaceWire network [8].

It is possible to employ one single SoCWire Switch and connect it to the arbitrary numbers of nodes or connect the SoCWire Switches together to build different topologies. Figure 1 shows an example of SoCWire architecture network with 9 nodes and 5 SoCWire Switches. The SoCWire switch uses wormhole routing and a time-slot based round robin mechanism. When a data frame is sent, the destination port in the switch is determined from the header. After that, the destination header is deleted and the remainder of the packet is transferred to the output port. In wormhole switching packets are split into smaller units of flow control called flits. In a subsequent routing step, after deleting the current destination header, the next flit in the header contains the destination address for the next stage. The SoCWire has a scalable data word width to support medium to very high data rates and each flit has two parts: the data word and one data control flag. The data control flag indicates if the current character is a data (0) or control character (1). SoCWire packet level is highly flexible and it is possible to implement a wide range of protocols. Figure 2 shows the format of a packet which contains Destination Address, Payload and End Of Packet/Error End of Packet (EOP/EEP) flits. The Destination Address can be one or more than one flits depending on the network topology and it is required to send packets over a SoCWire network to a specific target. The Payload contains the user data. A normal packet is completed with an EOP marker. The EEP marker indicates an erroneous packet which will be rejected by the target.

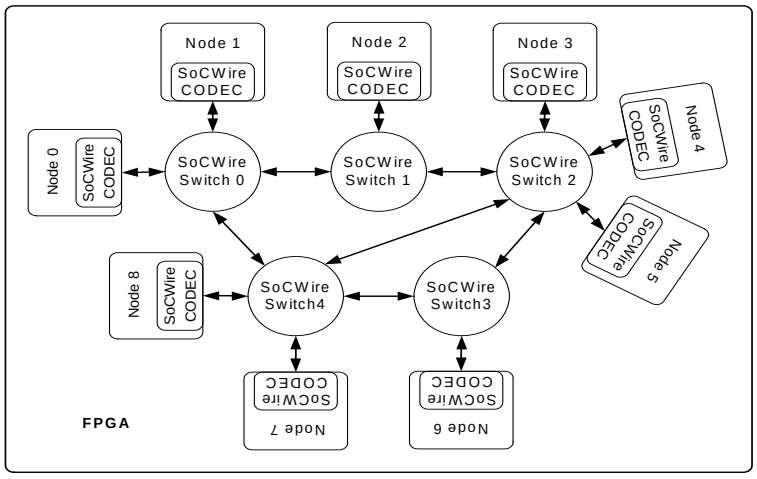

Figure 1. Example of an SoCWire architecture network

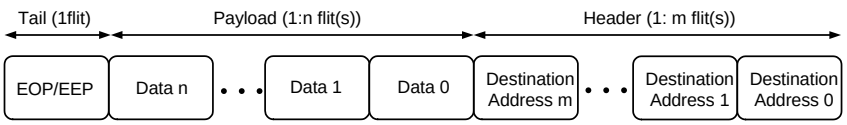

Figure 2. The packet format for SoCWire

The SoCWire NoC has two main components: *the SoCWire CODEC and the SoCWire Switch.* The SoCWire CODEC connects a node or host system to the SoCWire network. Figure 3 shows the SoCWire CODEC structure which has five main modules: *state machine*, *receiver*, *receiver FIFO*, *transmitter* and *transmitter FIFO*.

The State machine controls the operation of the SoCWire CODEC responses to errors and user requests. The receiver decodes the link data and sends the decoded user data to the receiver FIFO. All read operations are operated by this module. Link errors are detected and forwarded to the state machine. The receiver FIFO receives the user data from the receiver and transfers it to the host interface. It checks the outstanding counter and controls the Flow Control Token (FCT) handling. FCT is a control character and when space for eight more data characters in the receive buffer is available then a FCT is transmitted to allow eight data characters to be transferred. The receiver has a dual port RAM to store received data. The

transmitter FIFO transmits the user data from the host interface to the transmitter. It controls the credit flow and checks the credit counter to send the appropriate amount of user data. A credit-based flow control is implemented to avoid buffer overflows and cases data loss [8].

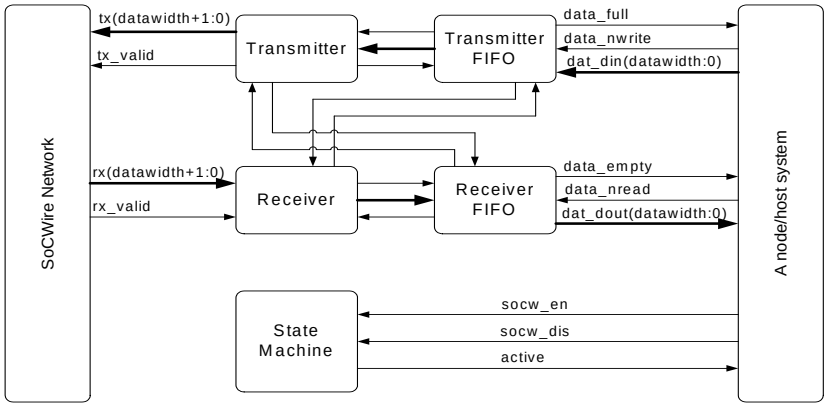

Figure 3. The SoCWire CODEC

Figure 4 shows the structure of an SoCWire Switch with six ports. It enables the transfer of packets arriving at one link interface to another link interface on the switch. SoCWire Switch has two main modules: *internal SoCWire CODECs* and a *routing switch*. SoCWire CODECs inside the switch are the same number of ports and the routing switch has an entrance unit that analyzes incoming packets, verifies the validation of the header, processes header deletion and passes the packet to another unit called the matrix. The matrix is a crossbar and manages the payload transfer to the destination port using cell modules. Each cell module represents an interconnection between two ports.

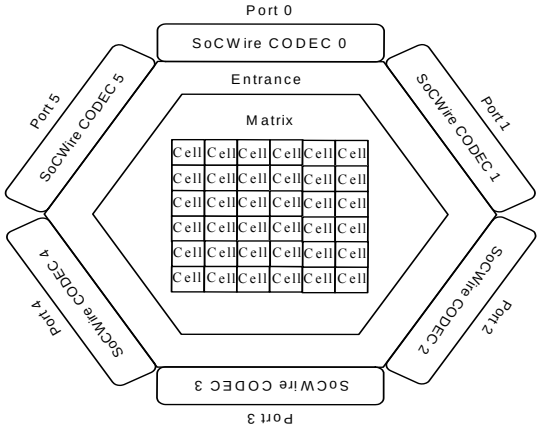

Figure 4. A SoCWire Switch with 6 ports

2.3. Motivation

This sequel presents the motivation for the extension of the SoCWire Switch to the SoCWire Router. A NoC Routing algorithm can be implemented as source or distributed routing algorithm. In source routing, the source node computes the path addresses and stores it in the packet header. The header which contains the path addresses uses bandwidth because it must be transmitted through the network. In distributed routing, the packet header only contains the destination address and each router computes the next output link address to move the packet towards its destination. Algorithmic routing uses a combinational logic circuit that computes the output port to be used as a function of the current and destination nodes and the status of the output ports [27].

Figure 5 depicts an example which shows the H.264 decoder with low resolution (h.264.dl) application task graph which has been extracted from reference [28] and Figure 6 shows a 4×4 mesh network on which the H.264 is mapped on. In this scenario, each node sends one packet with 7 flits data, 32 bits each flits. We consider a routing switch which has a header, data and end-of-packet flit and the header contains one address flit or a number of path addresses flits for consequent addressing inside the mesh network. There are 22 packet transactions inside the network which translates into 22×7=154 data flits. The required number of header flits can be calculated for each transaction with the formula:

$$\text{Number of header flits} = d(i,j) + 1; \quad (1)$$

Which $d(i,j)$ is the distance between router i and router j:

$$\mathrm{d\ (i,j)} = |\ \mathrm{X\ (d)}\ -\ \mathrm{X(c)}\ |\ + |\mathrm{Y(d)}\ -\ \mathrm{Y\ (c)}\ |; \quad (2)$$

And the sum of the required numbers of flits for all 22 transactions is 118. In addition, each packet has one end-of-packet flit which is 22 end-of-packets in overall. Therefore, 140 flits are required as the header addresses and tails to transfer 154 data flits which represents an overhead of 47.62%.

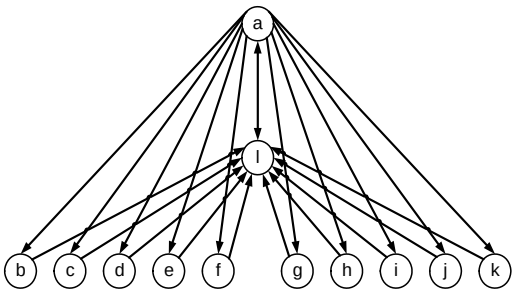

Figure 5. The h.264.dl task graph

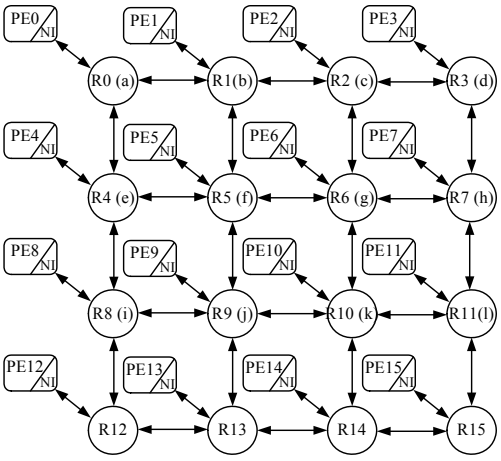

Figure 6. A 4×4 mesh network which this applicant is mapped on this network

On the other hand if we design a router which uses a combinational logic circuit to compute the output port to be used as a function of the current and destination nodes and the status of the output ports, there is only one flit needed as the header to route from any source to any destination. In this scenario, for all 22 transactions, 22 flits as destination addresses and 22 flits as end-of-packets are needed. Therefore, 44 flits are required header addresses and tails to transfer 154 flits of data which means that the overhead is reduced to 22.2%.

3. PROPOSED CONFIGURABLE ROUTER

The SoCWire Router has the standard features available as part of SoCWire and the SpaceWire standards. Figure 7 shows the configurable SoCWire Router architecture. This router has five main components: *internal SoCWire CODECs*, *Entrance*, *Matrix*, *Cells* and *input buffers*.

The SoCWire Router CODEC is a circuit that interfaces asynchronously with another CODEC located in the processing node. It is the same as the original SoCWire CODEC. The SoCWire Router has an address which is at maximum equal to 32 bits. When the data arrives inside the router ports via CODECs from local port(s) or buffers from communication ports, the entrance unit analyses incoming packets and makes routing decisions by comparing the packets header to the router address using a routing algorithm and passes the packet to the SoCWire Router matrix which is a crossbar to send the packet to the destination port. The routing algorithm varies depending on the configuration but in this work, it is derived from the X-Y routing algorithm which is a fixed shortest-path routing algorithm and has many advantages such as simplicity and power-efficiency. There is no deadlock scenario or the need to reorder flits. The data width of

all ports is set to 33 bits, 32 data bits and 1 control bit. The control bit indicates if the current character is a data or control character.

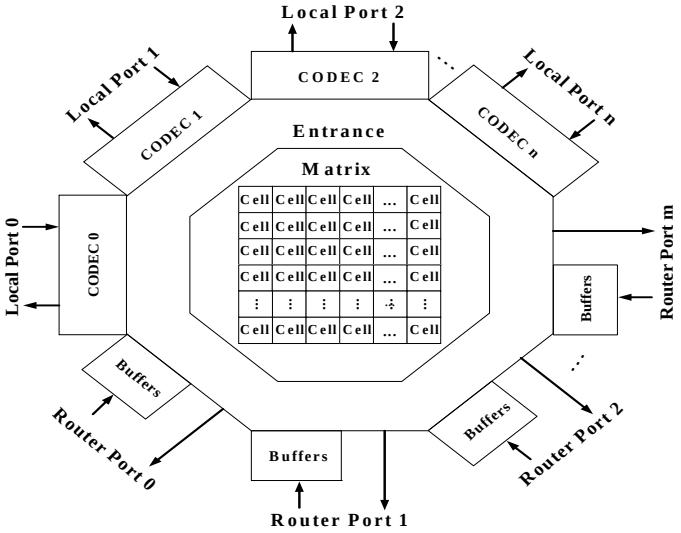

Figure 7. The Configurable SoCWire Router architecture

The packet structure used by SoCWire Router is shown in Figure 8 that reduces the size of the header which is a destination address to 33 bits. The minimum flits for each packet is three when only one data flit is present.

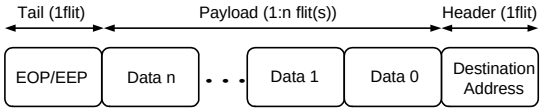

Figure 8. The packet format for SoCWire Router

This router can be configured with different number of local and communication ports to build 2D and 3D networks. For example, one possible configuration can be a SoCWire Router with five input and output ports (local, north, east, south and west) which has only one SoCWire CODEC in the local input port and buffers in communication ports. We call this configuration, base SoCWire Router. Each port can send/receive data to/from other ports concurrently and in cases where two or more ports want to send data to a port, a round robin arbiter is employed to avoid data collisions ensuring that only one sender has access to the port.

The Xilinx XC5VLX110T has been selected in this work for the implementation. We have selected bufferless communication ports for base SoCWire Router and in section 5 we will investigate the affect of the adding buffers of different sizes on performance and required area. Table 1 displays the base SoCWire Router utilization summary. The small area occupied by the SoCWire Router means that most of the device resources remain available to the user logic. The results are based on an implementation frequency of 100 MHz which is been used in all the system configurations to normalize the clock rate.

Table 1. Base SoCWire Router Utilization Summary

Logic Utilization	Used	Available	Utilization
Slice Registers	424	69120	0.61%
Slice LUTs	1277	69120	1.85%
Block RAM/FIFOs	1	148	0.68%

An initial experiment has evaluated the best (non-blocking) and worst (blocking) cases for the base SoCWire Router to evaluate its performance. In the best case, the SoCWire Router is tested so that there is no blocking taking place at any of the output channels. When the inputs arrive to the five input ports of the router, the router is able to service them at the same time and send them out to the output ports. The data from the local port is routed to the east port, data from the north port goes to the south port, data from east port is sent to the west port, data from south port goes to the local port and the data from west port is sent out to the north port simultaneously. The worst case is that all the packets arrive at the different input ports and request the same output port. One output port cannot service all of them simultaneously and this leads to blocking.

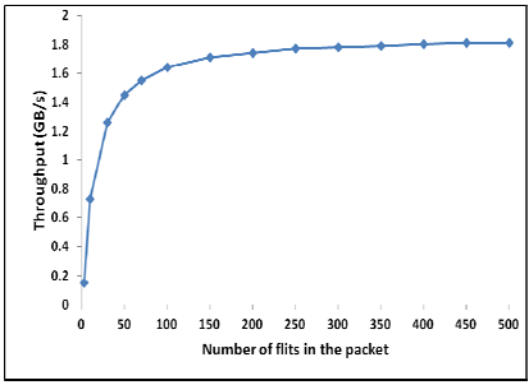

Figure 9. Base SoCWire Router throughput for non-blocking case

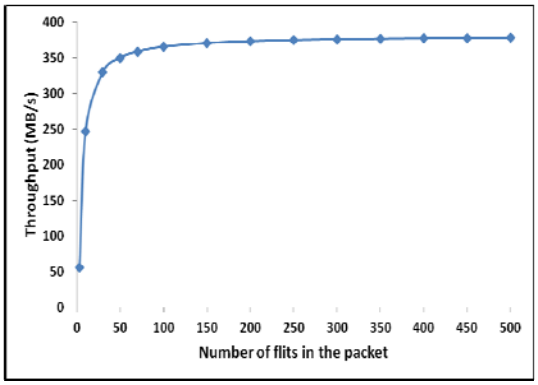

Figure 10. Base SoCWire Router throughput for blocking case

Figure 9 and Figure 10 shows the throughput for the tested non-blocking and blocking cases packets ranging in size from 3 to 500 flits. The overall trend shows that throughput increases with the rise in the number of flits with saturation clearly evident as packet sizes reached 50 flits and 150 flits for the worst and best case respectively. The performance of the base SoCWire Router is 378 Mbytes/s and 1.81 Gbytes/s for the worst and best cases respectively. The average latency increases linear with the number of flits in the packet. In addition, there is a linear relationship between the average latency and number of flits in the packet. The slope of the line is 2 and 10 for best and worst cases, respectively.

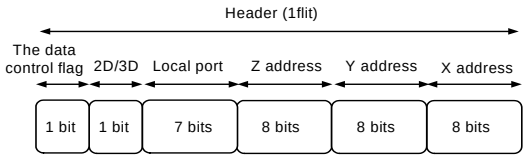

Figure 11. The header format of the SoCWire Router

There are two types of configuration for the SoCWire Router: the SoCWire Router with varying number of local ports for 2D and 3D networks. The SoCWire Router has the generic parameters to configure the number of local ports as well as the number of the network dimension type (2D/3D) and the network topology selection between mesh and torus to implement. Figure 11 shows the header flit of the SoCWire Router which is the destination address. This flit is 33 bits and has 6 parts: X, Y and Z dimensions of the destination address which are 8 bits each, local port number address which is 7 bits, dimension type (2D/3D) which is 1 bit to clarify the dimension of the networks and 1 bit the data control flag. The 2D/3D bit value is equal to “0”, the X and Y sections are used to determine the destination router address inside the network and the local port section determines the related destination local port when the network is 2D. In the 3D networks scenario, the 2D/3D bit value is equal to “1”, the X, Y and Z sections are used to determine the destination router address inside the network and the local port section determines the related destination local port.

3.1. SoCWire Router with Varying Number of Local Ports for 2D Networks

It is possible to configure the SoCWire Router with an arbitrary number of local ports for 2D networks although in this research we have limited the research to five characteristic router configurations: the base SoCWire Router which has one local port, SoCWire Routers with 2, 4 and 8 local ports for the 2D mesh topology and SoCWire Router with 1 local port for the 2D torus topology. We have employed the TRANC [29] routing algorithm which is a novel systematic approach for designing deadlock-free routing algorithms for torus NoCs. Figure 12 displays the routing pseudo code for the SoCWire Router with two local ports for 2D mesh network which is very similar for the SoCWire Routers with 4 and 8 local ports.

```
ΔX= destination router address X – current router X
ΔY= destination router address Y – current router Y
If ΔX==0 && Δy==0 && destination local port ==0
direction= Local port 0
If ΔX==0 && Δy==0 && destination local port ==1
direction= Local port 1
else if ΔX>0
direction= East
else if ΔX<0
direction= West
else if ΔY>0
direction= South
else if ΔY<0
direction= North
```

Figure 12. The routing pseudo code for the SoCWire Router with two local ports for 2D mesh network

Table 2. The Multi local ports SoCWire Router for 2D networks utilization summary

Logic Utilization	2D-Mesh-1 LP	2D-Mesh-2LP	2D-Mesh-4 LP	2D-Mesh-8 LP	2D-Torus-1 LP
Slice Registers	424	765	1458	3000	1235
Slice LUTs	1277	2166	3671	7761	2461
Block RAM/FIFO	1	2	4	8	1

Table 2 shows the utilization summary for the SoCWire Routers which are configured with 2, 4 and 8 local ports for 2D mesh and 1 local port for 2D torus topology when the buffer sizes in the communication port are equal to zero. This table reveals that by adding local ports the occupied area increases linearly.

3.2. SoCWire Router with Varying Number of Local Ports for 3D Networks

We have created two different 3 dimensional (3D) SoCWire Routers with one and two local ports. The 3D SoCWire Routers employ the XYZ routing algorithm.

```
ΔX= destination router address X – current router X
ΔY= destination router address Y – current router Y
ΔZ= destination router address Z – current router Z
If ΔX==0 && ΔY==0 && ΔZ==0
direction= Local port
else if ΔX>0
direction= East
else if ΔX<0
direction= West
else if ΔY>0
direction= South
else if ΔY<0
direction= North
 else if ΔZ>0
direction= Up
else if ΔZ<0
direction= Down
```

Figure 13. The routing pseudo code for the 3D SoCWire Router with one local port

Figure 13 displays the routing pseudo code for the 3D SoCWire Router with one local port and Table 3 reveals the utilization summary of implementing 3D SoCWire Routers with 1 and 2 local ports when the buffer sizes in the communication port are equal to zero. This table shows the 3D router with 1 local port occupies 65.23% more LUTs than the base SoCWire Router with one local port and the 3D SoCWire Router with two local ports consumes 28.86% more LUTs than the one with one local port.

Table 3. The 3D SoCWire Router Utilization Summary

Logic Utilization	3D-Mesh- 1LP	3D-Mesh- 2LP
Slice Registers	572	928
Slice LUTs	2110	2719
Block RAM/FIFO	1	2

4. EQUIVALENT NETWORKS AND PERFORMANCE EVALUATION PLATFORM

This section presents seven equivalent networks based on different SoCWire Router configurations and introduces our on-chip performance evaluation platform which is designed to measure the equivalent networks performance under a synthetic and realistic traffic loads.

4.1. Equivalent networks

In this sequel, the configured SoCWire Routers with 1, 2, 4 and 8 local ports and 3D SoCWire Routers with 1 and 2 local ports are used to build 7 equivalent networks with a common number of computing nodes of 16. Table 4 introduces the 7 equivalent networks specification with the resulting topologies shown in Figure 14 to Figure 20.

Table 4. Equivalent Networks Specification and Related Figure Numbers

Network	Dimension	Topology	Local ports	Figure
2D-Mesh-1LP	4×4×1	Mesh	1	Figure 14
2D-Mesh-2LP	2×4×1	Mesh	2	Figure 15
2D-Mesh-4LP	2×2×1	Mesh	4	Figure 16
2D-Mesh-8LP	1×2×1	Mesh	8	Figure 17
3D-Mesh-1LP	4×2×2	Cube	1	Figure 18
3D-mesh-2LP	2×2×2	Cube	2	Figure 19
2D-Torus-1LP	4×4×1	Torus	1	Figure 20

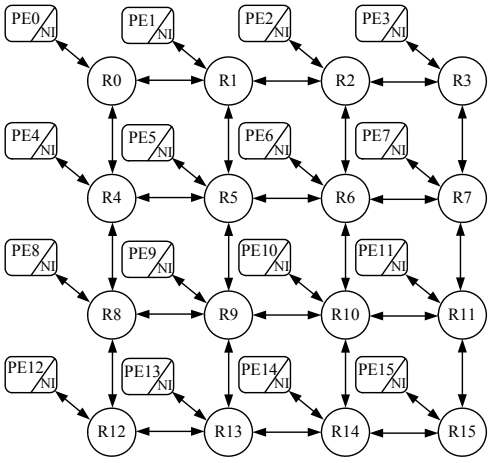

Figure 14. 2D-mesh-1LP

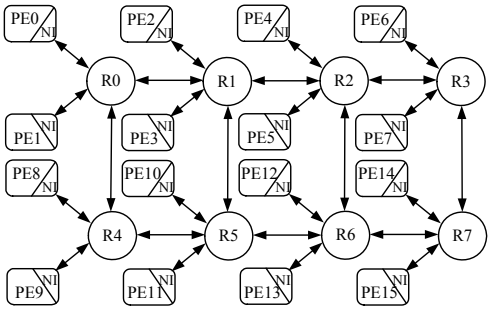

Figure 15. 2D-mesh-2LP

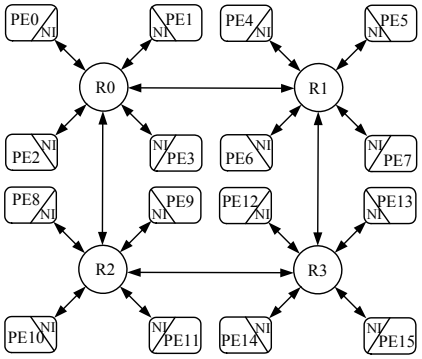

Figure 16. 2D-mesh-4LP

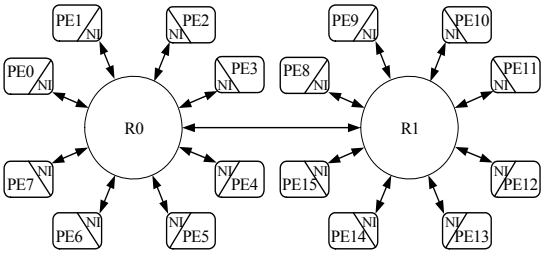

Figure 17. 2D-mesh-8LP

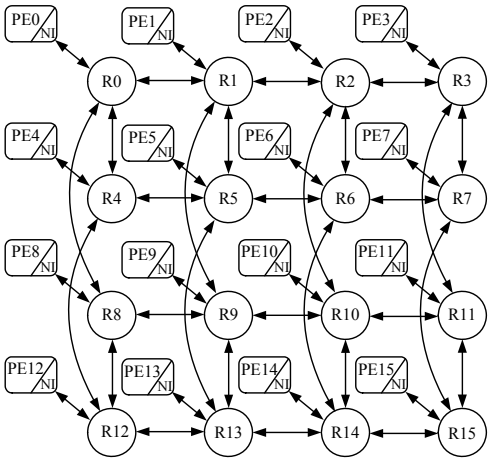

Figure 18. 3D-mesh-1LP

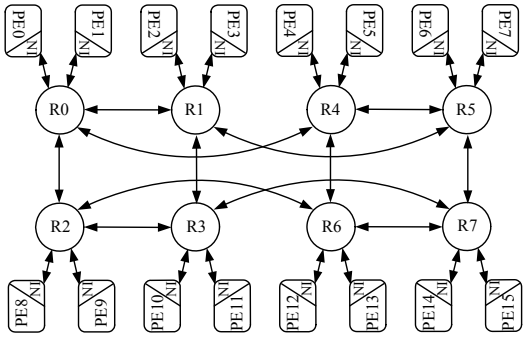

Figure 19. 3D-mesh-2LP

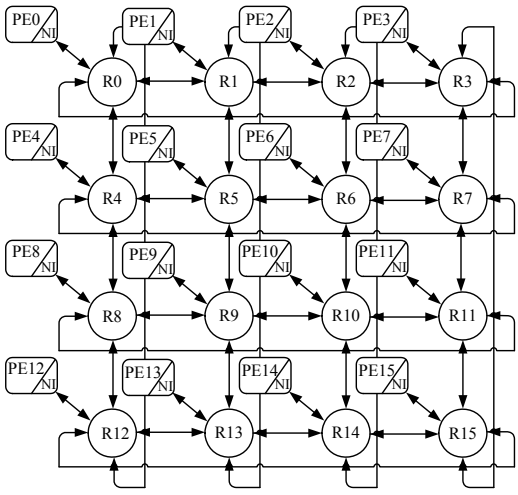

Figure 20. 2D-torus-1LP

4.2. NoC Performance Evaluation Platform

We have designed and implemented a NoC performance evaluation platform on the FPGA to verify and evaluate the created networks around the Leon3 SoC available in the GRLIB IP Library [30]. The Leon3 is a SPARC-compatible softcore processor which is developed by AeroFlex-Gaisler and interfaces to the AMBA bus architecture. The IP blocks for the AMBA bus, Leon3 processor and other SoC peripherals are

available in the GRLIB IP library. An additional IP block added to the GRLIB library is the reconfiguration controller [31] which is used to load new store bitstreams in external DDR memory into the tiles without host intervention via the ICAP hardware unit (Internal Configuration Access Port) which allows direct access to device fabric. This is the reasonable approach to follow since the FPGA device does not have enough on-chip memory to do this and the available on-chip memory is required for the data processing itself. This external memory provides a good trade-off between memory size, transfer overheads and on-chip resource utilisation.

Figure 21 presents the architecture of this performance evaluation platform. We have considered the SoCWire Network as the dynamic part to partially reconfigure the different equivalent networks for evaluation and the rest of the system as the statically configured. This allows the communication interconnect to map as a single block to a centralized area of the device which is connected to the evaluation platform. This approach is effective at using the current design flows of partial reconfiguration since it is possible to use the available FPGA resources optimally. If the number of local/communication ports change in a router the wiring infrastructure will change as well and this can be achieved by letting the P&R FPGA tools manage the resources in the assigned communication area without imposing excessive constraints.

Our performance evaluation system is based on different traffic types representing synthetic and realistic application traffic patterns. The evaluation platform has two phases: initialization and execution.

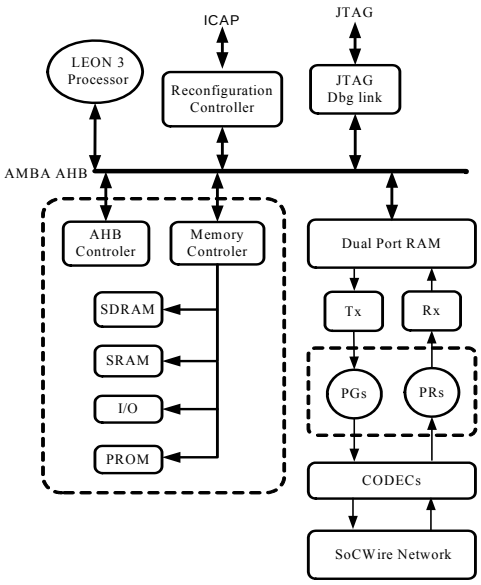

Figure 21. The performance evaluation platform for SoCWire networks

4.2.1. Initialization Phase

In the initialization phase, for the random traffic pattern case, Leon3 writes 32 bit words to the dual port RAM. This 32 bit word contains 16 bits to activate the Packet Generators (PGs), 8 bits to define the number of packets which should be generated by the PGs and 8 bits to determine the size of these packets in flits. The transmitter (Tx) reads the initialization word and sends it to each PG. If the traffic pattern belongs to a real application, Leon3 writes 64 bits to the dual port RAM for each transaction between 2 nodes. These 64 bits contain sequence number, execution time, dependencies, source and destination node addresses and data size. Sequence number determines the order of commands execution for each node. Execution time is the computation time required for a node to provide processed data to the network, dependencies identify the required data which should be received from one or several nodes to start the computation, source and destination address determine the sender and receiver address. Packet size shows the size of data in flits which is going to be sent. Transmitter (Tx) reads all of the initialization commands and sends them to the related Packet Generators (PGs). In this step, PGs are initialized and the execution phase starts.

4.2.2. Execution phase

In the execution phase, if the traffic pattern is a random traffic, PGs start to generate frames. Each frame has a header which is a flit and determines the destination address, 1 or more data flits depending on the initialization and a tail flit to clarify the EOP. The destination address is generated by a random number

generator which is a Linear Feedback Shift Register (LFSR) counter [32]. When a PG generates and sends a packet, it waits for a certain amount of time to generate and send the next packet. These timing intervals between sending packets can be fixed or randomly generated. When the traffic pattern is real application traffic, PGs start to generate frames based on the commands located in the dual port RAM. The frames format is the same as the random traffic frames.

In addition; each packet has a start sending time stamp and source node address as well. A CODEC connects a PG to the SoCWire Router. When a node receives a packet, it will pass the packet to the Packet Receptor (PR) through the CODEC. The PR compares the destination address in the header of the received packet to the node address and if they are the same, PR will extract the source address and sender timing stamp. PR passes the source and destination address and source and destination timing stamp to the Receiver (Rx). Rx receives all the information from PRs and writes them in the dual port RAM. Finally, Leon3 reads the execution results which have been written in the dual port RAM for further processing and reconfigure the dynamic part with another network bitstreams for evaluation. Table 5 shows the complexity of the LEON3 and our evaluation platform.

Table 5. LEON3 and Performance Evaluation Platform Complexity

Logic Utilization	LEON3 system	Evaluation Platform
Slice Registers	4505	20402
Slice LUTs	8963	34097
Block RAM/FIFOs	21	48

5. EVALUATION OF THE MESH NETWORK WITH DIFFERENT BUFFER SIZES IN THE COMMUNICATION PORTS

In this section, we are going to evaluate the 2D-Mesh-1LP network with different buffer sizes in the SoCWire Router communication ports.

5.1. Performance

Performance is analyzed in terms of throughput and latency after implementing the system in the target board running at a normalized frequency of 100MHz. The throughput shows the efficiency of delivering packets in the network and depends on topology and routing policy which is XY for all the 2D topologies and XYZ for the 3D topologies to be able to perform a fair comparison. The time required for traversing the network is referred to as the latency of a network. The average latency is the mean of the latencies of all received packets in the topology [33].

We have created a dynamic area in the device to map the 2D-Mesh-1LP network with different buffer sizes in the communication ports and have investigated the performance with four different synthetic traffic patterns. The rest of the evaluation platform is statically configured.The resulting device layout is shown in Figure 22 with the central area occupy by the communication network clearly shown. The size of the partial bitstream for the communication network is 1.7 Megabyte for the larger configuration.

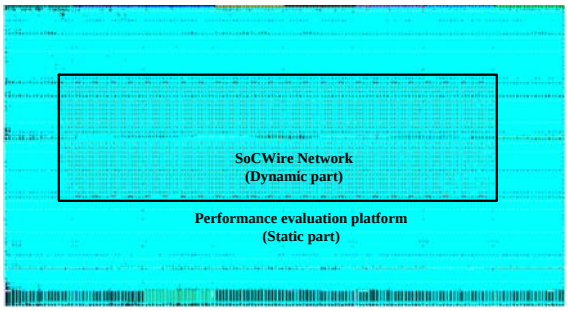

Figure 22. The resulting device layout with the central area occupy by the communication network

5.1.1. Uniform traffic pattern

The first traffic pattern considered is the uniform traffic pattern. In this pattern, each node sends 100 packets, with 15 flits each, to random targets which are the other nodes. There is a random timing interval between the sending packets which indicate the average time between data transfer requests. We have

evaluated the networks with different ranges of random firing intervals. A lower value indicates a more heavily loaded NoC.

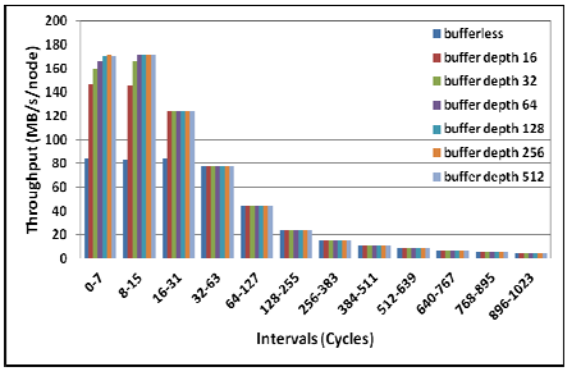

Figure 23. The throughput for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the uniform traffic pattern

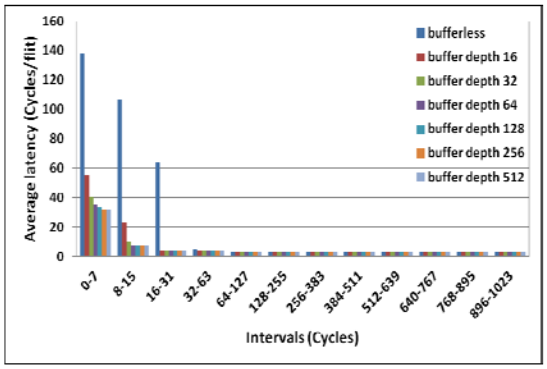

Figure 24. The average latency for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the uniform traffic pattern

As it can be seen in the Figure 23 and Figure 24 which are the throughput and average latency for this traffic pattern, adding the buffers to the communication ports helps to increase the SoCWire network throughput and decrease the average latency in a heavy traffic pattern when the intervals are 0-63 cycles. This trend changes after 64 cycles in which the buffers in the communication ports do not help the performance and bufferless communication ports has a better average latency. In addition, the size of added buffer to the communication ports has a direct effect on the networks performance when there is a heavy traffic pattern especially when the intervals are 0-15 cycles. The numbers in the figures (i.e., 16, 32, 64, 128 and 512) show the depth of the communication ports buffer with a width constant of 32 bit.

5.1.2. One Hot Spot Traffic Pattern

The second traffic pattern is a concentrated node traffic pattern with 1 hot spot. In this traffic pattern, nodes 0-14 send 100 packets, 15 flits each, to the target node which is node 15. Similar to the first pattern, there are random timing intervals between the sending of packets. Figure 25 and Figure 26 show the throughput and average latency for this traffic pattern. Overall the bufferless configuration decreases the average latency and adding buffers to the communication ports does not help to improve the performance.

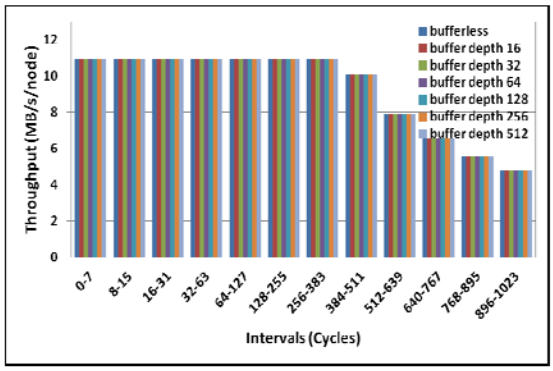

Figure 25. The throughput for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the one hot spot traffic pattern

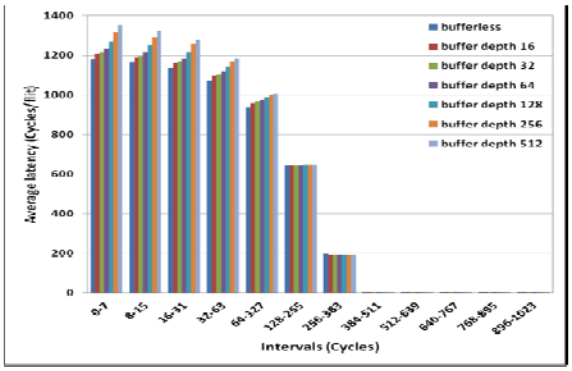

Figure 26. The average latency for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the one hot spot traffic pattern

5.1.3. Two Hot Spots Traffic Pattern

The third traffic pattern is a concentrated node traffic pattern with 2 hot spots.

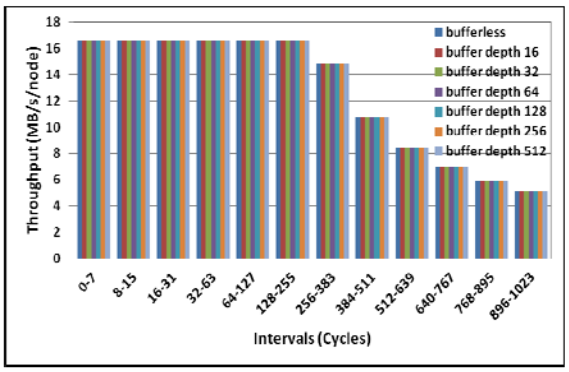

Figure 27. The throughput for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the two hot spots traffic pattern

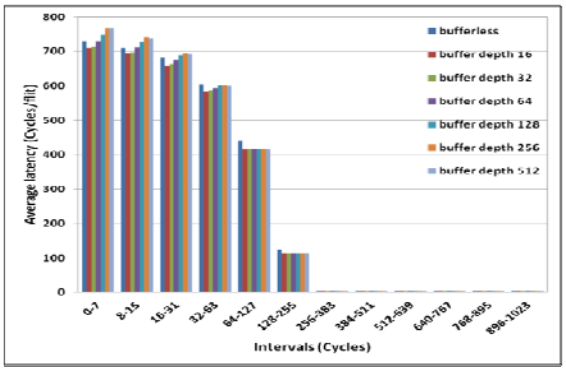

Figure 28. The average latency for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the two hot spots traffic pattern

In this traffic pattern, nodes 0-15 sends 100 packets, 15 flits each, to the target nodes which are nodes 0 and 15. There are random timing intervals between the sending of packets like the other traffic patterns. Figure 27 and Figure 28 which are the throughput and average latency for this case, reveal adding a small size of the buffers(16 and 32) to the communication ports helps to reduce the average latency when the traffic pattern is heavy and medium(0-383 cycles intervals) but does not change the overall throughput.

5.1.4. Unloaded Traffic Pattern

The fourth traffic pattern is an unloaded traffic pattern. In this traffic pattern, node 0 sends 100 packets, 15 flits each, to the random target nodes which are nodes 1-15. There are random timing intervals among the sending of packets. Figure 29 and Figure 30 display the throughput and average latency for this traffic pattern. Figure 29 shows adding buffers to the communication ports increase the throughput in the heavy traffic scenario when the intervals are 0-15 cycles. Figure 30 reveals when the traffic is not heavy and the intervals are larger than 16 cycles, the bufferless reduce the average latency.

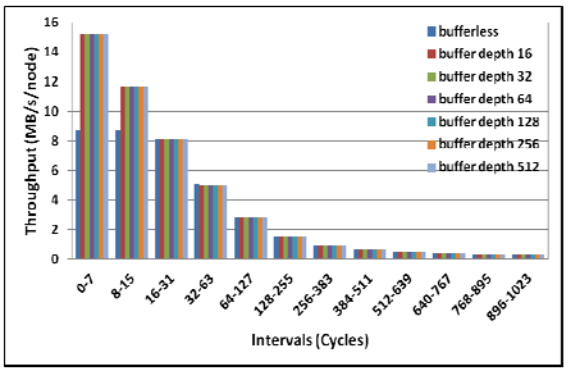

Figure 29. The throughput for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the unloaded traffic pattern

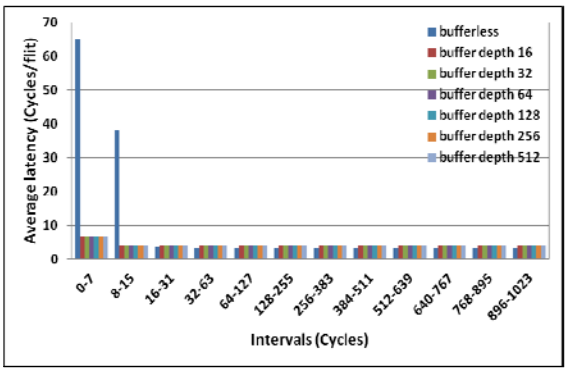

Figure 30. The average latency for the 2D-Mesh-1LP network with different buffer sizes in the communication ports with the unloaded traffic pattern

5.2. Complexity

Table 6 displays the utilization summary for the 2D-Mesh-1LP with seven different buffer sizes in the communication ports. As it can be seen in this table, 4x4-bufferless network decreases the occupied LUTs

and block of RAM/FIFOs by 30.1% and 33.3% respectively compare to the 512 depth buffer scenario. Adding depper buffers to the communication ports increases the occupied area and requires more Block RAM/FIFOs. For example, using 512 depth buffer instead of 16 increases the occupied area by 8.16%.

Overall, the performance evaluation results reveal that adding the buffers to communication ports help the SoCWire Router networks performance when the intervals are 0-63 cycles and the traffic pattern is a heavy traffic. When the traffic is not a heavy traffic pattern, adding buffers does not help. In addition, adding buffers increase the complexity at least by 30%. We have choosen the bufferless to explore in the next section because although buffers help the performance in a heavy traffic pattern, they increase complexity considerable.

Table 6. The Utilization Summary for the 2D-Mesh-1LP with Different Seven Buffer Sizes in the Communication Ports

Networks	Slice Registers	Slice LUTs	Block RAM/FIFOs
4x4-bufferless	6064	13678	16
4x4-buffer depth 16	10913	18046	48
4x4- buffer depth 32	11188	18238	48
4x4- buffer depth 64	11425	18366	48
4x4- buffer depth 128	11681	18814	48
4x4- buffer depth 256	11964	18823	48
4x4- buffer depth 512	12193	19518	48

6. EVALUATION OF THE EQUIVALENT NETWORKS

In this section we have created a dynamic area in the device to map the communication network and have investigated network performance under different topology configurations. The rest of the evaluation platform is statically configured. The resulting device layout is shown in Figure 31 with the central area occupy by the communication network clearly shown. The size of the partial bitstream for the communication network is 1.4 Megabytes for the larger configuration.

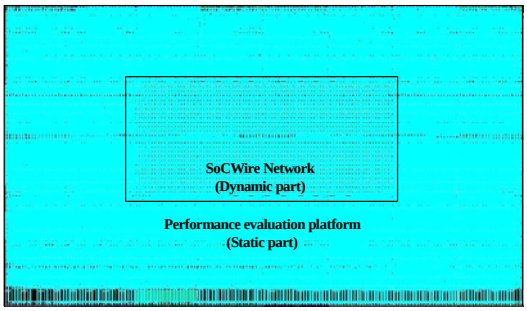

Figure 31. The resulting device layout with the central area occupy by the communication network

6.1. Performance

We have considered five different traffic patterns to analyse the performance of the different network configurations. The first four are synthetic traffic patterns and the fifth is a realistic traffic pattern. We have considered a single implementation frequency which is 100MHz to investigate the efficiency of the different topologies at transferring data bypassing implementation effects that generate different maximum frequencies.

6.1.1. Uniform Traffic Pattern

The first traffic pattern considered is the uniform traffic pattern. In this pattern, each node sends 100 packets, with 15 flits each, to random targets which are the other nodes. There is a random timing interval between the sending packets. We have evaluated the networks with different ranges of random firing intervals.

Figure 32 and Figure 33 display the throughput and average latency for this scenario. As it can be seen in these figures, the torus network is the best choice for 0-63 cycles intervals and the 2D-Mesh-4LP for 64-1023 cycles intervals to increase the throughput and reduce the average latency compared to the other

networks. The worst case performance corresponds to the 2D-Mesh-8LP due to the high congestion on the single link between the two routers.

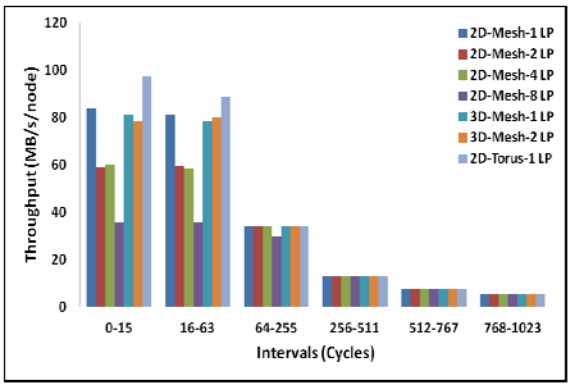

Figure 32. The networks throughput for the uniform traffic pattern

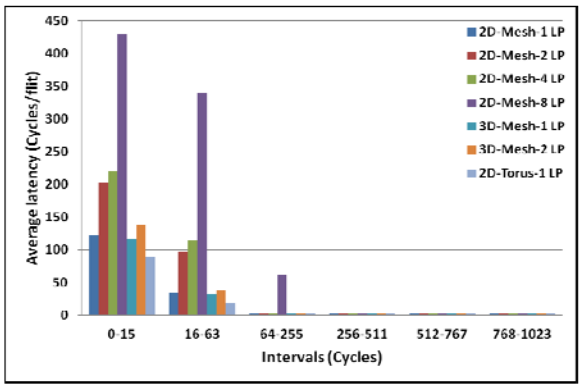

Figure 33. The networks average latency for the uniform traffic pattern

6.1.2. One Hot Spot Traffic Pattern

The second traffic pattern is a concentrated node traffic pattern. In this traffic pattern, nodes 0-14 send 100 packets, 15 flits each, to the target node which is node 15. Similar to the first pattern, there are random timing intervals between the sending of packets.

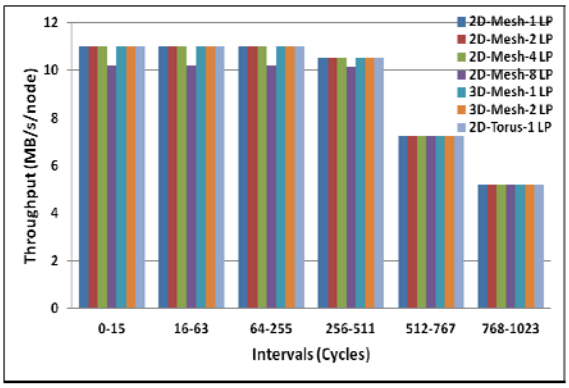

Figure 34. The networks throughput for the one hot spot traffic pattern

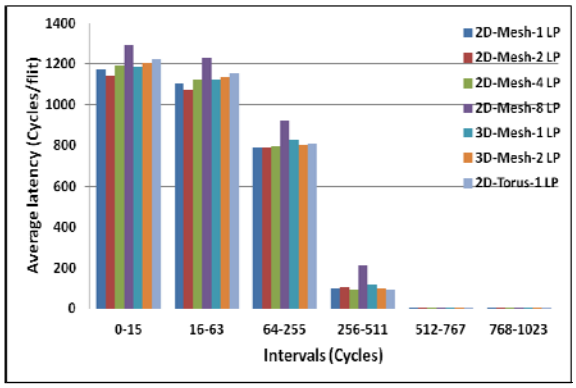

Figure 35. The networks average latency for the one hot spot traffic pattern

Figure 34 and Figure 35 show the throughput and average latency for this traffic pattern. In this scenario, the best choice is the 2D-Mesh-2LP for 0-63 cycles intervals and the 2D-Torus-1LP, 2D-Mesh-4LP and 3D-Mesh-2LP for 64-1023 cycles intervals. The worst choice is the 2D-Mesh-8LP for both throughput and latency.

6.1.3. Two Hot Spots Traffic Pattern

The third traffic pattern is a concentrated node traffic pattern with 2 hot spots. In this traffic pattern, nodes 0-15 sends 100 packets, 15 flits each, to the target nodes which are nodes 0 and 15. There are random timing intervals between the sending of packets like the other traffic patterns. Figure 36 and Figure 37 show the throughput and average latency for this traffic pattern. Figure 37 reveals that the 2D-Torus-1LP, 3D-Mesh-2LP and 2D-Mesh-4LP are the best options to reduce the average latency compared to the other networks. The 2D-Mesh-8LP is the worst choice with the highest latency.

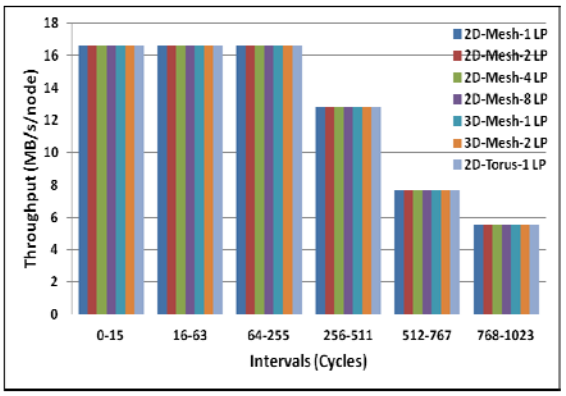

Figure 36. The networks throughput for the two hot spots traffic pattern

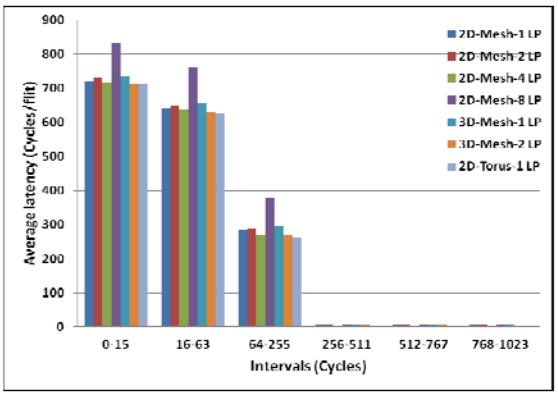

Figure 37. The networks average latency for the two hot spot traffic pattern

6.1.4. Unloaded Traffic Pattern

The fourth traffic pattern is an unloaded traffic pattern. In this traffic pattern, node 0 sends 100 packets, 15 flits each, to the random target nodes which are nodes 1-15. There are random timing intervals among the sending of packets. Figure 38 and Figure 39 display the throughput and average latency for this traffic pattern. These figures show that the torus network and 2D-Mesh-4LP offer the best performance and the lower latency compared to the other networks. The worst choices for both throughput and latency are the 2D-Mesh-1LP and the 3D-Mesh-1LP.

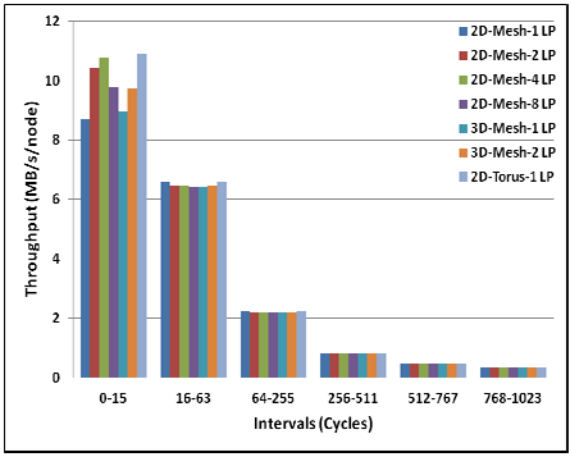

Figure 38. The networks throughput for the unloaded traffic pattern

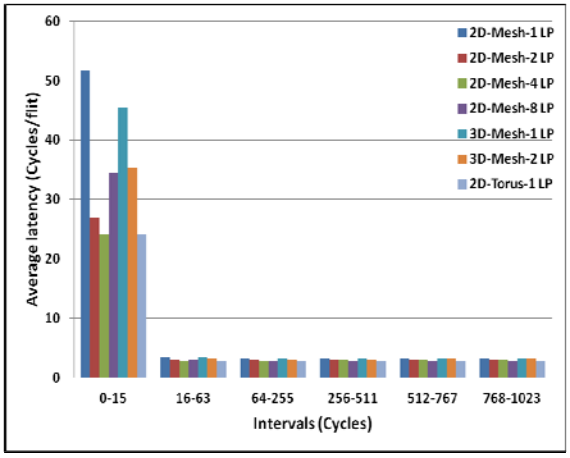

Figure 39. The networks average latency for the unloaded traffic pattern

6.1.5. Realistic Application Traffic Pattern

As a real case, we have employed a realistic traffic pattern which is a H.264 decoder with low resolution (h.264.dl) application from the MCSL (Multi-constraint system-level) benchmark suite to compare the networks [28]. The MCSL contains a set of traffic patterns based on real applications that have been adjusted to the make them compatible with the evaluation platform. We have separated the communication between and computation in the nodes and only used the communication between the nodes.

Figure 40 and Figure 41 display the normalized throughput and average packet latency when the clock rate of the network and computing nodes is 100MHz and Figure 42 and Figure 43 show the normalized throughput and average packet latency when there are two different frequency islands for the network and computing nodes which are 100 and 500MHz respectively under h.264.dl application. Figure 40 and Figure 42 show that the normalized throughput for this realistic application is largely constant and independent of the topology with a small benefit for the 2D-Torus-1LP, 2D-Mesh-4LP and 2D-Mesh-8LP.On the other hand

Figure 41 and Figure 43 show that the 2D-Torus-1LP and 2D-Mesh-4LP networks offer a significant reduction in latency and are the best choices.

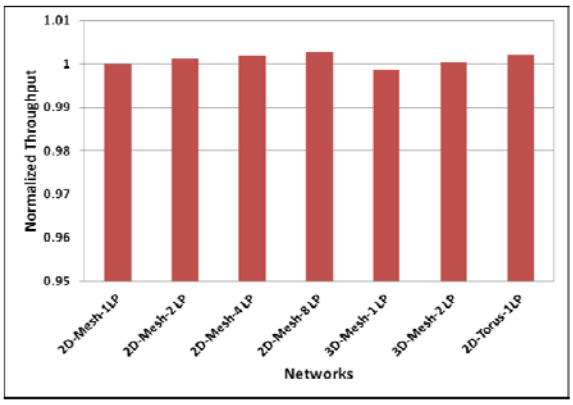

Figure 40. Normalized throughput for mesh network under h.264.dl real application (network and computing nodes frequency=100MHz)

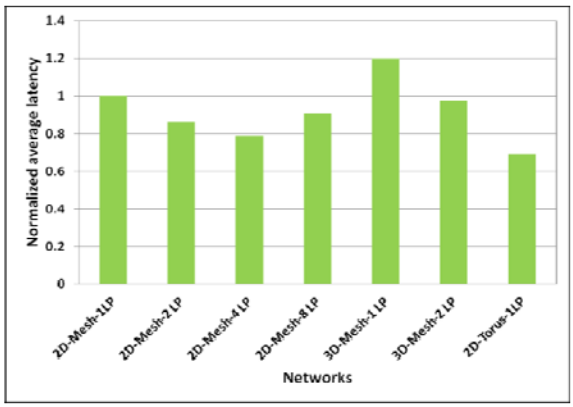

Figure 41. Normalized average latency for mesh network under h.264.dl real application (network and computing nodes frequency=100MHz)

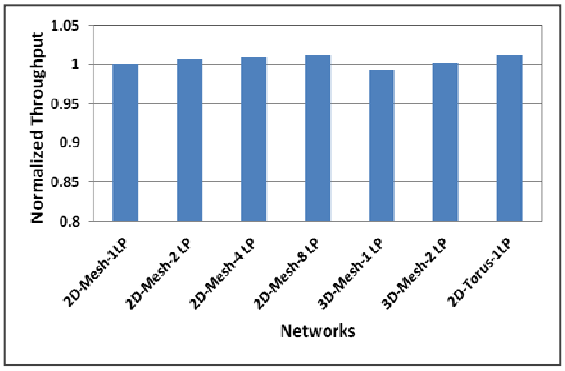

Figure 42. Normalized throughput for mesh network under h.264.dl real application (network frequency=100MHz and computing nodes frequency=500MHz)

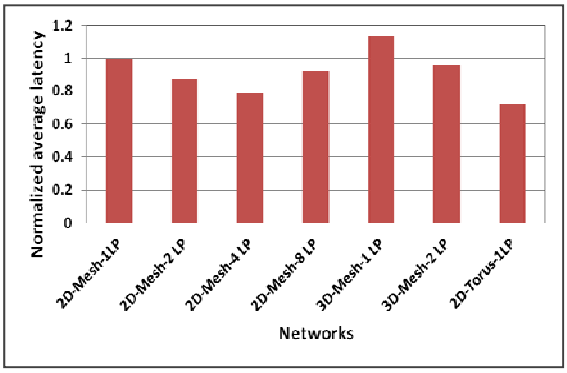

Figure 43. Normalized average latency for mesh network under h.264.dl real application (network frequency=100MHz and computing nodes frequency=500MHz)

The traffic pattern for this application and many other realistic applications generate low traffic in the network due to the processing time needed in the computing nodes before a result packet is ready to be injected in the network. These means that the timing intervals between the sending of packets tend to be quiet large. Overall, the experiment reveals that conclusions found in the literature based on synthetic and randomized traffic patterns should be re-examined with traffic profiles obtained from real applications.

6.2. Complexity and Implementation Results

We have measured the occupied area of each network separately to compare. In addition, the XPower tool, which is the Xilinx power analysis tool, has been used to estimate the consumed power of the networks. We have considered the hierarchy power, which includes DCM power, BRAM power, signal power and logic power, to compare the networks. Table 7 shows the complexity of the equivalent networks, consumed power and maximum operation frequency of the networks. This table reveals that if the objective is to optimize the NoC from an area point of view the 2D-Mesh configuration with 4 local ports is the most area efficient while the 2D-Torus with 1 local port is the most area intensive. The torus is a popular topology that reduces congestion in the central part of the network delivering good performance but it produces a high overhead in terms of resources utilization when implemented on FPGAs as seen in the table. In addition, this table reveals that for the fixed number of the nodes, adding the local ports to the SoCWire Router and 3D SoCWire Router increases the consumed power. Therefore, the power consumption has a direct relationship

with the local ports in the SoCWire Router and the regular mesh network with 1 local port SoCWire Router, consumes less power than the 3D and torus networks while the 3DMesh-2LP and 2D-Torus-1LP have similar power requirements. The maximum operation clock frequency for the networks is highly dependent on the selection of the network area in the FPGA. We have examined this by changing the dynamic area to different places. In this scenario, adding more local ports or multidimensional decrease the maximum achieved clock frequency. The torus network achievable clock frequency is degraded due to the need to create long links between the border nodes but in this case is 6.4% reduction compared to the 2D-Mesh-1LP due to the size of the mesh network and mapping the networks are mapped as a single block.

Table 7. The Complexity, Power Consumption and Maximum Achieved Clock Frequency for the Networks

Networks	Power (mW)	Max. Clock frequency (MHz)	Slice LUTs	Slice Registers	Block RAM/ FIFOs
2D-Mesh-1LP	50.58	120.584	13678	6064	16
2D-Mesh-2 LP	51.78	110.314	10976	5400	16
2D-Mesh-4 LP	53.47	104.646	**10908**	**5243**	16
2D-Mesh-8 LP	63.26	102.208	11147	5351	16
3D-Mesh-1 LP	53.34	111.086	15444	6728	16
3D-Mesh-2 LP	53.75	107.631	11657	5774	16
2D-Torus-1LP	56.24	112.854	16680	6784	16

Table 8. The Evaluation Results Summary

Traffic pattern	Intervals (Cycles)	Performance	Area	Power
Uniform	0-63	2D-Torus-1LP	**2D-Mesh-4LP**	2D-Mesh-1LP
Uniform	64-511	**2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
Uniform	512-1023	**2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
1-hot-spot	0-63	2D-Torus-1LP, **2D-Mesh-4LP** ,3D-Mesh-2LP	**2D-Mesh-4LP**	2D-Mesh-1LP
1-hot-spot	64-511	2D-Torus-1LP, **2D-Mesh-4LP** ,3D-Mesh-2LP	**2D-Mesh-4LP**	2D-Mesh-1LP
1-hot-spot	512-1023	2D-Torus-1LP, **2D-Mesh-4LP** ,3D-Mesh-2LP	**2D-Mesh-4LP**	2D-Mesh-1LP
2-hot-spots	0-63	2D-Torus-1LP, **2D-Mesh-4LP** ,3D-Mesh-2LP	**2D-Mesh-4LP**	2D-Mesh-1LP
2-hot-spots	64-511	2D-Torus-1LP, **2D-Mesh-4LP** ,3D-Mesh-2LP	**2D-Mesh-4LP**	2D-Mesh-1LP
2-hot-spots	512-1023	2D-Torus-1LP, **2D-Mesh-4LP** ,3D-Mesh-2LP	**2D-Mesh-4LP**	2D-Mesh-1LP
Unloaded	0-63	2D-Torus-1LP, **2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
Unloaded	64-511	2D-Torus-1LP, **2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
Unloaded	512-1023	2D-Torus-1LP, **2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
h.264.dl	0-63	2D-Torus-1LP, **2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
h.264.dl	64-511	2D-Torus-1LP, **2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP
h.264.dl	512-1023	2D-Torus-1LP, **2D-Mesh-4LP**	**2D-Mesh-4LP**	2D-Mesh-1LP

After performing an analysis of the observed performance and complexity results it is possible to conclude that the 2D-Mesh-8LP is not a good choice in general. The best overall performance is achieved by the Torus configuration but the complexity costs are high. The complexity of the 2D-Mesh-2LP is approximately 20% lower than the traditional 2D-Mesh-1LP but its performance is also lower specially for the uniform traffic case. The 3D configurations perform similarly among them but the complexity of the 3D with 1 local port is considerably higher than the equivalent with 2 local ports. Overall, the 2D-Mesh-4LP offers better level of performance to the traditional 2D-Mesh-1LP with lower complexity so it could prove a suitable choice for FPGA-based NoCs built around the SoCWire standard. Table 9 summarizes the evaluation results in function of how heavily loaded the NoC is (intervals), the traffic patterns expected in the application, occupied are and power consumption.

7. CONCLUSION

This paper has presented the configurable SoCWire Router as an extension to the SoCWire NoC which is a robust, fault tolerant and reconfiguration friendly NoC. The base SoCWire Router uses 1277 LUTs when it is configured with 1 local and 4 communication ports for the mesh topology. The performance of this router in the best and worst cases is 1.81 Gbytes/s and 378Mbytes/s respectively with a clock frequency of 100MHz. A number of SoCWire Routers with varying number of local ports for 2D and 3D networks have been investigated from a performance, power and area points of view. To thoroughly test the proposed configurations a performance evaluation platform based on the Gaisler Leon3 processor has been developed. The evaluation platform implements a number of equivalent networks with 16 processing nodes with partial reconfiguration technique. The equivalent networks are evaluated with five different traffic patterns to

investigate the ideal network and the ideal configuration for the SoCWire Router. The evaluation results show that the selection of the ideal router is different for traffic patterns based on synthetic or real applications. The selection of the ideal router also depends if the design goal is to optimize for power, area, performance or a combination of these but overall, the mesh network with four local ports router is a good candidate to employ instead of the traditional mesh network with one local port router. In addition, adding buffers to communication ports helps the network performance when there is a heavy traffic pattern but it does not help in other cases and significant more resources. These conclusions indicate that the partial reconfiguration features available in modern FPGAs could be used to deploy different interconnects at run-time depending on active application and design goal. Further work will consider more realistic traffic patterns derived from real applications and implementation effects under hard area and performance constraints. The current work is based on a Xilinx V5 LX110T that with 69K logic cells does not offer enough resources to build larger systems but with new FPGAs such as the latest Xilinx Virtex-7 with millions of logic cells, it will be possible to create very large communication networks with hundreds of processing elements and study the scalability of this method in these cases. This is part of our future work.

REFERENCES

[1] Osterloh B, Michalik H, Habinc SA, Fiethe B. *Dynamic Partial Reconfiguration in Space Applications*. NASA/ESA Conference on Adaptive Hardware and Systems (AHS). San Francisco, California. 2009; 1: 336-343.

[2] Jongman Kim. A Comprehensive Approach to Design Network-On-Chip Architectures for Soc/Multicore Systems. Ph.D. Dissertation. Pennsylvania State University, University Park, PA, USA. AAI3266141. 2007.

[3] Benini L, De Micheli G. Networks on chips: a new SoC paradigm. *Computer*. 2002; 35(1): 70-78.

[4] Dally WJ, Towles B. *Route packets, not wires: on-chip interconnection networks*. Proceedings Design Automation Conference. 2001; 1:684- 689.

[5] Rijpkema E, et al. *Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip.* IEE Proceedings Computers & Digital Techniques. 2003; 150(5): 294-302.

[6] Lee H G, et al. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. *ACMTransactions on Design Automation of Electronic Systems.* 2007; 12(3): 1-20.

[7] Osterloh B, Michalik H, Fiethe B. *SoCWire: A Robust and Fault Tolerant Network-on-Chip Approach for a Dynamic Reconfigurable System-on-Chip in FPGAs*. 22nd International Conference on Architecture of Computing Systems(ARCS'09). Delft, Netherlands. 2009; 50-59.

[8] *System-on-Chip Wire, source codes and documents.* IDA, 5 May 2009. Available: www.socwire.org. [Accessed 25 December 2012].

[9] *ECSS, Space Engineering: SpaceWire-Links, nodes, routers, and networks*. ESA-ESTEC, Noordwijk, Netherlands. 2008. ECSS-E-50-12C.

[10] Koch D, Haubelt C, Teich J. *Efficient Reconfigurable On-Chip Buses for FPGAs*. Field-Programmable Custom Computing Machines, (FCCM '08). Palo Alto, California. 2008; 287-290.

[11] Möller L, et al. *A NoC-based infrastructure to enable dynamic self reconfigurable systems.* 3rd International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC'07). Montpellier, France. 2007.

[12] Moraes F, et al. HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip. *INTEGRATION, The VLSI Journal*. 2004; 38(1): 69-93.

[13] Bobda C, Ahmadinia A. Dynamic Interconnection of Reconfigurable Modules in FPGA. *IEEE Design & Test of Computers-Special Issue Networks on Chip.* 2005; 2(25): 443-451.

[14] Bobda C, et al. *DyNoC: A dynamic infrastructure for communication in dynamically reconfugurable devices*. International Conference on Field Programmable Logic and Applications (FPL'05). Tampere, Finland. 2005; 153-158.

[15] Jovanovic S, et al. *CuNoC: A Scalable Dynamic NoC for Dynamically Reconfigurable FPGAs*. Field Programmable Logic and Applications (FPL'07). Amsterdam, The Netherlands. 2007; 753-756.

[16] Pionteck T, et al. *Communication Architectures for Dynamically Reconfigurable FPGA Designs*. IEEE International Parallel and Distributed Processing Symposium (IPDPS'07). 2007; 1-8.

[17] Rana V, et al. *A Reconfigurable Network-on-Chip Architecture for Optimal Multi-Processor SoC Communication.* 16th Annual IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Rhodes, 2008; 321: 326.

[18] Hubner M, et al. *Run-time reconfigurable adaptive multilayer network-on-chip for FPGA-based systems.* IEEE International Symposium on Parallel and Distributed Processing (IPDP). Miami, Florida. 2008:1-6.

[19] Modarressi M, Tavakkol A, Sarbazi-Azad H. Application-Aware Topology Reconfiguration for On-Chip Networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.* 2011; 19(11): 2010-2022.

[20] Spacewire_light. Available: http://opencores.org/project. [Accessed 25 December 2012].

[21] Spacewire. Available: http://opencores.org/project. [Accessed 25 December 2012].

[22] SpaceWire Switch Core. Available: http://www.spacewire.co.uk. [Accessed 25 December 2012].

[23] SpaceWire products and service. Available: http://www.4links.co.uk/spacewire-products. [Accessed 25 December 2012].

[24] ATMEL, 2008, AT7910E. Available: www.atmel.com/dyn/resources/ prod_documents/doc7796.pdf [Accessed 25 December 2012].
[25] GRSPWROUTER SpaceWire Routing Switch. Available: http://www.gaisler.com/ . [Accessed 25 December 2012].
[26] Osterloh B, et al. *Architecture verification of the SoCWire NoC approach for safe dynamic partial reconfiguration in space applications*. NASA/ESA Conference on Adaptive Hardware and Systems (AHS). Anaheim, California. 2010; 1-8.
[27] Flich J, Rodrigo S, Duato J. *An Efficient Implementation of Distributed Routing Algorithms for NoCs*. International Symposium on Networks-on-Chip – NOCS. Newcastle upon Tyne, UK. 2008; 87-96.
[28] Weichen Liu, et al. *A NoC Traffic Suite Based on Real Applications*. IEEE Computer Society Annual Symposium on VLSI (ISVLSI). Chennai, India. 2011: 66-71.
[29] Rahmati D, et al. Power-efficient deterministic and adaptive routing in torus networks-on-chip. *Microprocessors and Microsystems - Embedded Hardware Design.* 2012; 36(7): 571-585.
[30] *Leon3 Processor*. Available: http://www.gaisler.com/ . [Accessed 25 December 2012].
[31] Nabina A, Nuñez-Yañez JL. *Dynamic Reconfiguration Optimisation with Streaming Data Decompression*. Field Programmable Logic and Applications (FPL'10). Milano, Italy. 2010: 602-607.
[32] George M, Alfke P. *Linear Feedback Shift Registers in Virtex Devices*. Xilinx application note Xapp210.
[33] Dally, William J, Towles B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, Inc, 2004.

BIOGRAPHIES OF AUTHORS

Arash Farhadi Beldachi received his B.Sc. in Electrical and Electronic engineering from Azad University, Saveh, Iran, in 2001, and his M.Sc. in Electronic engineering from Iran University of Science & Technology (IUST), Tehran, Iran, in 2007. He is currently a Ph.D. student in the Department of Electrical and Electronic engineering at Bristol University, UK, under supervision of Dr. Jose Luis Nunez-Yanez, working on Reconfigurable Network-On-Chip (RNoC).

Mohammad Hosseinabady received the B.S. degree in electrical engineering from the Sharif University of Technology, Sharif, Iran, in 1992, the M.S. degree in electrical engineering and the Ph.D. degree in computer engineering from the University of Tehran in 1995 and 2006, respectively. He is currently a research fellow with the University of Queen's University, Belfast, working System-on-Chip Architectures and Programmable Systems (SoC).

Jose Luis Nunez-Yanez received the B.S. degree from Universidad de La Coruna, La Coruna, Spain, and the M.S. degree from Universidad Politécnica de Cataluña, Barcelona, Spain, in 1993 and 1997, respectively, both in electronics engineering, and the Ph.D. degree in the area of hardware architectures for high-speed data compression from Loughborough University, Loughborough, U.K., in 2001. During 2005 he worked at STMicroelectronics, Italy after receiving a Marie Curie fellowship in the area vector architectures for video processing and in 2010 he worked ARM Ltd, Cambridge with a Royal Society fellowship in the area of system-level energy estimation and modelling. He is currently a Senior Lecturer with the Department of Electronic Engineering, University of Bristol, U.K., His current research interests include the areas of data and video compression, reconfigurable and energy efficient computing and brain modelling.