Network Interface Design and Implementation for NoC on FPGA with advanced Hardware and Networking Functionalities

Veeraprathap.V, M.Nagaraja, M.Z.Kurian

Abstract— As we are living in a billion transistor era, the number of components on a given chip increases drastically, System on Chip (SoC) architectures become even more powerful. Key to this architecture is the ability to integrate multiple heterogeneous components into a single architecture, which requires modularity and abstraction. An integral part of this architectural design is the methods by which the various components communicate with one another. Network on Chip (NoC) architectures attempt to address these concerns by providing various component level architectures with specific interconnection network topologies and routing techniques. Networks-On-Chip (NoCs) have been proposed as a promising replacement to eliminate many of the overheads of buses and MPSoCs connected by means of general-purpose communication architectures. This paper presents the design and implementation of FPGA based Network on chip (NoC) which is scalable packet switched architecture with advanced Networking functionalities such as store & forward transmission, error management, power management and security. All these features are built on basic NI core, which includes data packetization/depacketisation, frequency conversion, data size conversion and conversion of protocols with limited circuit complexity and cost

Index Terms— Intellectual Property (IP), Multi-Processor System-on-Chip (MPSoC), Network-on-Chip (NoC), Network-Interface (NI), VLSI Architecture.

I. INTRODUCTION

Network on chip (NoC) [1] is an evolving design technology which is used for growing a packet switched communication infrastructure, which contains hundreds of IP cells, connected on a single Multi-processor System on Chip (MPSoC) [2]. NoCs affords a design methodology for interconnect architecture for relating hundreds of IP cores which can be used for general purpose processors, application specific processors, digital signal processors and so on. Network interface (NI) will be considered as the key element of NoC, which makes IP macrocells to be associated to on-chip communication backbone in plug and play fashion. These NI's are also considered as building blocks of the NoC. Basically NI's takes care of data packetization/ depacketisation to and from the NoC; it encodes all the packet header and promises a successful end to end data delivery between the IP cores.

A NoC packet comprises of a header and a data payload and they are splitted into units called as flits. And all these flits are routed in the same path across the Network.

Manuscript received on June, 2013.

Mr.Veeraprathap.V, M.Tech 4Th Semester Student, Department of E&C, SSIT, Tumkur.

Dr.M.Nagaraja, Associate professor, Department of E&C, SSIT, Tumkur.

Dr.M.Z.Kurian, Dean & HOD, Department of E&C, SSIT, Tumkur.

Header field is composed of Network layer header whose content is find out by NI, according to the node map network configuration, and a transport layer header (TLH) which holds the information used by the NI's for end to end transaction management. Some of the researchers proposed that they can implement the conversion of the data size, protocol and frequency between the original IP bus and NoC. But the IP bus can be standardised one such as AMBA (Advanced microcontroller Bus architecture), AXI (Advanced eXtensible Interface) or OCP (Open Core Protocol) [4] or a custom bus such as ST Bus [5].The latest researches on the NI architecture focus on implementing more features to directly support advanced Networking functionalities, the challenge here in doing so is by keeping NI area, power and latency overhead as low as possible with respect to connected IP cores. But integrating all these above said features with limited circuit complexity will be difficult. To overcome these draw backs we have designed and implemented a FPGA based NoC which is scalable packet switched architecture with advanced Networking functionalities such as store & forward transmission, error management, power management and security.

II. RELATED WORK AND CONTRIBUTIONS

In this section, we provide a detailed analysis of various factors to be considered while designing and implementing NoC.

Topology

From the communication point of view, there have been several topologies for NoC architecture. These include mesh,butterfly, torus, ring, octagon and irregular interconnection networks [6], [7]. Various researchers have exploited these different NOC topologies for their NOC implementations. Kim et al. have used a star-based NOC that communicated using the principle of CDMA (Code Division Multiple Access) [8]; Here in our implementation we are using a Spidergon topology, (A ring one with an additional across link for each node to reduce network crossing latency).The Spidergon NoC is a network which is proposed by ST Microelectronics. Fig. 1 is an instance of Spidergon topology with 16 nodes. The small box with numbers stands for a node; every node is composed of a processing element (PE) and a router. The processing element can be memory, IP and so on; the router is used for routing packets from the local processing element to other routers. Each node connects to other nodes in three directions: clockwise, anticlockwise and cross-link. For example, the node 0 connects to node 1 in clockwise, to node 15in anticlockwise, and to node 8 in cross-link. The routing algorithm in the Spidergon is quite simple.

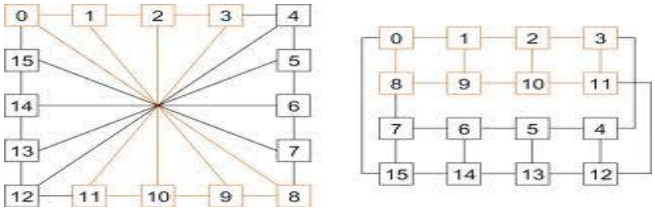

Fig 1: Spidergon Topology

Router Architecture

NOC architectures are designed to perform based on packet-switched networks. This has led to new and efficient principles for design of routers for NOC [8]. Assume that a router for the mesh topology has four inputs and four outputs from/to other routers, and another input and output from/to the Network Interface (NI). Routers can implement various functionalities - from simple switching to intelligent routing. Since embedded systems are constrained in area and power consumption, but still need high data rates, routers must be designed with hardware usage in mind. For circuit-switched networks, routers may be designed with no queuing (buffering). For packet-switched networks, some amount of buffering is needed, to provision bursty data transfers. Such data originate in multimedia applications such as video streaming. Buffers can be provided at the input, at the output, or at both input and output [8].

Routing protocols

Efficient routing of messages within the network is essential in order to fully exploit the power of the computing resources and achieve better performance for applications running on them. A good routing algorithm should not only provide low latency for messages but should also be deadlock free when the network is simultaneously routing multiple messages. Due to this border links of the region more heavily used as compared to other links. Adaptive routing is one solution that can reduce the problem of local congestions. Normally, the term adaptive refers to a possibility to sense congestion and take action to divert from it. When regions are used in a NoC it is possible that this information is incorporated in the routing algorithm so that occurrence of congestion is reduced or avoided.

Switching Techniques

Switching techniques can be categorized based on network characteristics. The Circuit switched networks reserve a physical path before transmitting the data packets; while packet switched networks transmit the packets without reserving the entire path. Packet switched networks can further be classified as Wormhole, Store and Forward (S&F), and Virtual Cut Through Switching (VCT) networks In Wormhole switching networks, only the header flit experiences latency. Other flits belonging to the same packet simply takes the path taken by the header flit. If the header flit is blocked then the entire packet is blocked. It does not require any buffering of the packet.

Therefore, the size of the chip considerably reduces. However, the major drawback of this switching technique is a higher latency. Thus, it is not a suitable switching technique for real-time data transfers. Al-Tawil et al. provided a well-structured survey of Wormhole Routing techniques and its comparison with other switching techniques [8].

S&F switching forwards a packet only when there is enough space available in the receiving buffer to hold the entire packet. Thus, there is no need for dividing a packet into flits. This reduces the overhead, as it does not require circuits such as a flit builder, a flit decoder, a flit stripper and a flit sequencer. The CLICHÉ implementation of a NOC is an example of store-and-forward switching [3]. Millberg et al. employed this switching technique in their Nostrum NOC implementation [8]. In VCT switching, a packet is forwarded to the next router as soon as there is enough space to hold the packet. However, unlike S&F, the VCT algorithm divides a packet into flits, which may be further divided into phits.

Flow Control

Flow control determines how network resources, such as channel bandwidth, buffer capacity, and control state, are allocated to a packet traversing the network. The flow control may be buffered or buffer less .The Buffer less Flow Control has more latency and less throughput than the Buffered Flow Control. The Buffered Flow Control can be further categorized into Credit Based Flow Control, ACK/NACK Flow Control and Handshaking Signal based Flow Control. In Credit Based Flow Control, an upstream node keeps count of data transfers, and thus the available free slots are termed as credits. Once the transmitted data packet is either consumed or further transmitted, a credit is sent back. Bolotin et al. used Credit Based Flow Control in QNOC.

In Handshaking Signal Based Flow Control, a VALID signal is sent whenever a sender transmits any flit. The receiver acknowledges by asserting a VALID signal after consuming the data flit. Zeferino et al. used handshaking signals in their SoC IN NOC implementation.

In the ACK/NACK protocol a copy of a data flit is kept in a buffer until an ACK signal is received. On assertion of ACK, the flit is deleted from the buffer; instead if a NACK signal is asserted then the flit is scheduled for retransmission.

Virtual Channel

The design of a virtual channel (VC) is another important aspect of NOC. A virtual channel splits a single channel into two channels, virtually providing two paths for the packets to be routed. There can be two to eight virtual channels. The use of VCs reduces the network latency at the expense of area, power consumption, and production cost of the NOC implementation. However, there are various other added advantages offered by VCs.

Network deadlock/live lock: Since VCs provide more than one output path per channel there is a lesser probability that the network will suffer from a deadlock; the network live lock probability is eliminated (these deadlock and live lock are different from the architectural deadlock and live lock, which are due to violations in inter-process communications).

Performance improvement: A packet/flit waiting to be transmitted from an input/output port of a router/switch will have to wait if that port of the router/switch is busy. However, VCs can provide another virtual path for the packets to be transmitted through that route, thereby improving the performance of the network. Supporting guaranteed traffic: A VC may be reserved for the higher priority traffic, thereby guaranteeing the low latency for high priority data flits [7], [8].provides an alternative path for data traffic, thus it uses the wires more effectively for data transmission. Therefore, we can reduce the wire width on a system (number of parallel wires for data transmission).

Reduced wire cost: In today's technology the wire costs are almost the same as that of the gates. It is likely that in the future the cost of wires will dominate. Thus, it is important to

use the wires effectively, to reduce the cost of a system. A virtual channel Units

For example, we may choose to use 32 bits instead of 64 bits. Therefore, the cost of the wires and the system will be reduced.

Buffer Implementation

A higher buffer capacity and a larger number of virtual channels in the buffer will reduce network contention, thereby reducing latency. However, buffers are area hungry, and their use needs to be carefully studied and optimized. Zimmer et al. and Bolotin et al. proposed a simple implementation of buffer architecture for NOC [4] they concluded that increasing the channel bandwidth is preferable to reducing the latency in NOC

III. DESIGN OF CORE NETWORK INTERFACE

A. Top level Architecture of NI.

In NOC interface IP cores are commonly classified as Master and Slave IPs. Initiator NIs are connected to Master IPs, which will convert the IP request transaction into NOC traffic, and also translates the packets received from NOC into IP response transactions. Similarly, Target NIs also exists, which are connected to slave IPs, here the target NIs represents a mirrored architecture where the requests are decoded from the NOC and the responses are encoded.

In both NI types, two main domains are identified Fig (2(a)) which shows the top view of an initiator NI. And Fig (2(b)) shows the top view of the target NI, here the shell is considered as IP specific, and Kernel in NOC specific, where each one has its own peculiar functionality.

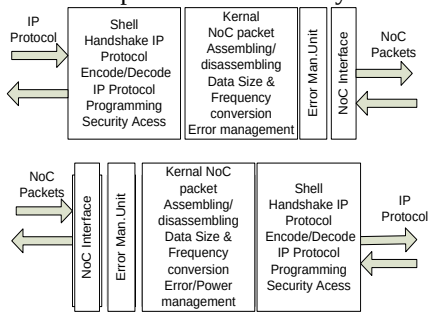

Fig 2: Top view of the NI design: (a) Initiator and (b) Target

Fig 2(a) and 2(b) highlights some advanced networking functionalities such as programming, security, error and power management. The main aim of the shell and Kernel separation is to abstract IP specific properties (such as bus protocol and data size) from NOC side properties.

In this manner the NOC becomes an IP-Protocol agnostic interconnect which includes protocols, bus size, clock frequency, the master or slave IP is using and all modules in the system may communicate with each other.

The conversion features should be implemented in two directions, one in request path which is from master to slave IPs and one more in response path from slave to master respectively. Here the Kernel and all its associated NOC interface is IP protocol independent and its common to all possible NI's. This supports the following IP bus protocols. AMBA, AXI, which is a de-facto standard in embedded systems ST-Bus type used by ST microelectronics, DNP a distributed network processor. Mainly used by ATMEL to build a multi-tile MPSOS architecture.

B. Kernel-Shell Interface by BI-Synchronous FIFO

The Kernel is interfaced to the shell by means of a FIFO like interface. As shown in fig 3, encoded data which in coming from the shell are stored in two FIFO's. One is header FIFO (which holds transport layer and network layer headers) and a payload FIFO (which holds bus raw data). Each FIFO consists of its own read and write manager which will updates FIFO pointers and status and provides frequency conversion mechanism. The Kernel is connected to the NOC interface through two additional FSMs. In the request path, an output FSM (OFSM) which will reads the headers and payloads and converts them to the packets according to the need of NOC protocol. Similarly in the response path, an Input FSM (IFSM) collects the packets and splits the header and payload flits into their respective FIFO's. Here it should be noted that the NI encodes both TLH and the NLH, while decoding it takes only TLH into account, because as already the packets have been reached the destination and the routing data is also not required. Bi-synchronous FIFOs are used in the NI scheme. The frequency conversion mechanism is accomplished between NOC and connected IP. Each read (write) FIFO manager re-synchronizes in its own clock domain, the pointer of the write (read) manager in the other clock domain, hence the status of the FIFO that is empty or full is find out using comparing synchronized pointers, and the header and the pay load FIFO's can be correctly managed. To increase the robustness of the synchronization pointers they adopt gray encoding. A FIFO location is sized according to the larger data size between data in and data out. A FIFO row is sized according to the smaller data size between data in and data out as shown in Fig 4.

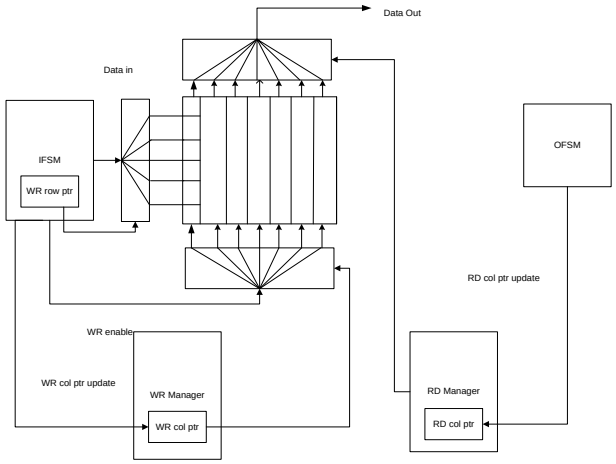

Fig 4: upsize conversion in single FIFO.

Up size conversion is accomplished by writing by rows and reading by columns and downsize conversion is exactly opposite to that. In particular cases when Shell / Kernel frequency conversion nor size conversion is not required, it is possible to remove the bi-synchronous FIFOs, by setting their size to zero, by this which will save area and power consumption this feature is known as Zero kernel FIFO.

IV. ADVANCED NETWORK INTERFACE FEATURES

A. Store and Forward (S & F)

Kernel FIFOs in the request and response paths contains flits, which are either received from the NOC and to be decoded towards the IP bus, or encoded from a bus transaction and to

be Transmitted over the interconnect. The default NI behavior is that as soon as flit is available from the NOC it is extracted as a result the original traffic at an interface (NOC or bus) has an irregular shape, as soon as store and forward is enabled, flits are kept into the internal Kernel FIFOs, until the whole packet is encoded /received and then they are transmitted / decoded all together. In this way an irregular traffic is converted to bubble free traffic thus improving overall system performance.

S & F can be enabled on both request path and response path independently, at NOC- to- bus level, it is possible to enable per-packet S & F, while several S & F options can be selected at NOC-to bus level. The mechanism of handling pre-packet S&F implementation is quite simple, after completion of a packet, the FSM controlling the FIFOs reading is in a state where only the header FIFO is checked , to extract the beginnings of the new packet. Here the concept to handle per-packet S & F, in both directions is to keep packet header hidden to the reading logic by simply not updating the header FIFO write pointer. When the entire packet is stored in the FIFO (both header and payload), the header is unmasked and made visible by updating the header FIFO pointer, and the reading logic detects the presence of a new packet.

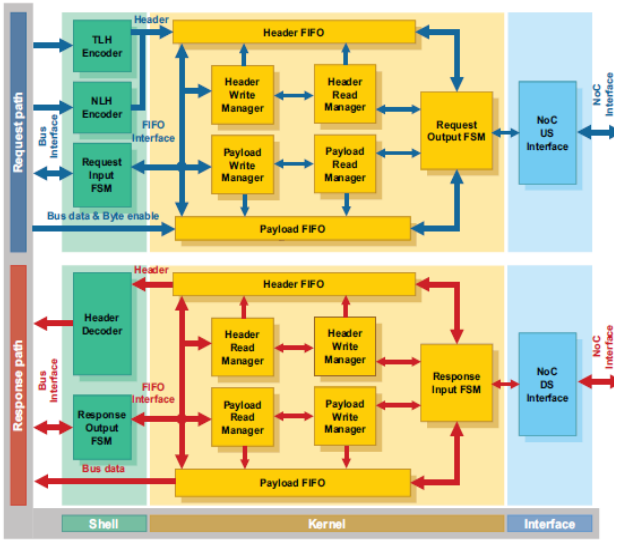

Fig 3: Main blocks in NI micro architecture

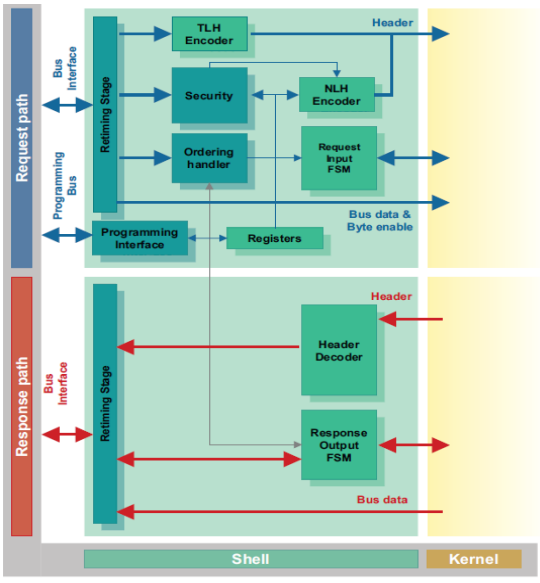

Fig 5: Advanced features in a NI initiator.

B. Error Management Unit (EMU)

The EMU is an optional stage which can be implemented between the Kernel and NOC interface. The behavior of the EMU is different for both initiator and target NIs. In an initiator NI the EMU can handle bad address errors and security violations. When the address of the master IP transaction is not in the range of assigned memory map, or when the transaction is trying to access the protected memory zone without having permission the packet in its header is flagged as an error packet.

In the later stages the EMU filters the packet directed to the NOC US-interface to avoid it to enter the network, and creates a response packet. Which remaps the request header on a new response header, and if required adds a dummy payload. The EMU also take care of properly managing the incoming traffic at the DS NOC interface during the power down mode. All the traffic which is received in request during power down mode is flushed by the EMU, so that it never reaches the slave IP.

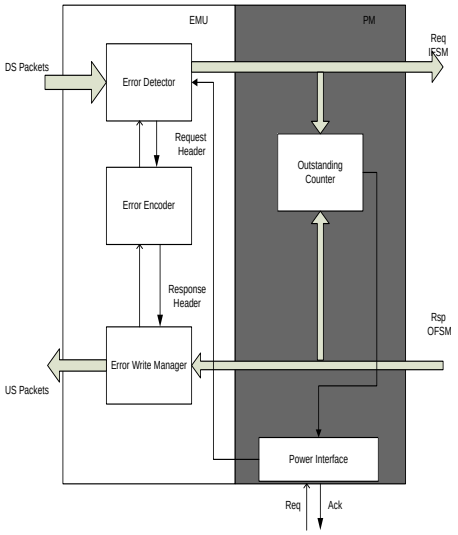

Fig 6: EMU and Power manager in Target NI.

The EMU itself generates an error response to the master originating the request. As shown in Fig 6, the EMU is composed of 3 blocks which are

Error detector, which flushes all error traffic. In Initiator NIs, the outgoing error traffic is identified by a flag in the header, while in Target NIs all incoming packets are flushed if the connected Slave IP is in power down mode;

Error encoder, which assembles a new NoC packet to be channeled in the response path;

Error write manager, which is basically a traffic-light to avoid simultaneous traffic to the US NoC Interface from Kernel Response and from the EMU in a Target NI, while in an Initiator NI it avoids interference between the DS NoC Interface and the EMU both trying to access the Kernel Response.

If a request packet does not contain an error, the EMU behaves transparently and does not add any clock cycles of latency.

C. Power Manager (PM)

This feature is available only for Target NIs connected to slaves which may be turned off to save power. The PM is always coupled to an EMU block which rejects incoming

NoC packets trying to access the Target NI when the connected slave IP is in power down mode as shown in fig 6. A simple req/ack protocol controls the power up/down state of the NI, by means of a dedicated interface: each request (req set to 1) acknowledged by the PM unit (ack set to 1) makes the NI power state switch from UP to DOWN and vice versa. It may happen that a request for power down is sent to the PM while the slave IP is still elaborating a number of pending transactions. In this case the Target NI stops accepting packets from the network and waits for all pending transactions to be processed before acknowledging the request and switching to power down mode. The power manager is a completely new feature introduced by the proposed NI.

D. Security

The security service, available only in NI Initiators, acts as a hardware firewall mechanism. It introduces a set of rules that transactions coming from the Master IP must satisfy to gain access to the network. The security rules involve:

Lists of memory intervals under access control;

Lists of Master IPs that may have access to a certain memory region;

Lists of access types.

Security rues are applied in the Security block during packet encoding. If a test fails the security check, the corresponding transaction is marked as an error in the NLH and it is detected by the EMU, which must be activated as well to properly manage security violations. The illegal packet is then discarded and does not consume network bandwidth, and the error response to the Master IP is directly generated by the EMU itself. The rules that allow a transaction to access the network are described by means of a security map. In this map a number of memory regions are defined, and associated to region IDs the same map defines how these regions can be accessed. Access to these zones can be allowed only to Master IPs belonging to specified groups. Source and for each protected region depending on the address and the Source, the memory access can be immediately allowed or immediately denied, or go through a security rule check. Naturally,

V. NOC SYNTHESIS AND SIMULATION RESULTS

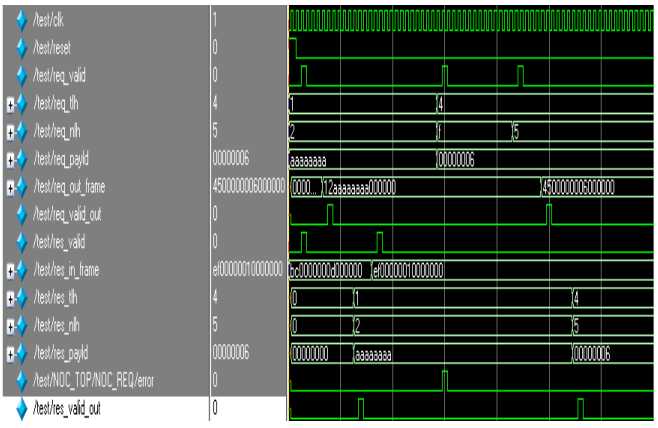

Fig 7: Simulation Result of NI interface.

As shown in synthesis result one set of input of 40 bits which is in the range assigned is given once, in one more set of input one variable is exceeding the range so an error signal is generated. And 64 bit output value is obtained at the output.

A. NI configuration space.

The Implemented NI is designed to support a wide configuration space, not only the advanced feasters can be enabled or disabled, also some of the basic characteristics can be configured like flit size, bus data size, pay load & header FIFO size, conversion of frequency and data size. By changing the configurations set different trade-offs between performance and complexity are achieved.

B. Verification and synthesis flow.

The exact functionality of the proposed NI design in multiple configurations has been verified at different abstractions levels. First a random functional verification environment has been created and applied to multiple configuration of HDL database.

The various building blocks of the NOC are exploited using Verilog HDL language and Xilinx ISE tool. Which is a software tool produced by Xilinx for synthesis and analysis of HDL designs, which enables the developer to synthesize (Compile) their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction, and it is simulated and debugged using modelsim and the simulation waveform can be obtained which will be further useful for analysis. The NI data base is verified and validated at different level by both synthesis and simulation. Finally, the design is configured on SPARTAN 3AN family FPGA kit with the help of Verilog HDL.

VI. CONCLUSION

A Novel advanced design of Network interface for on chip communication has been presented in this paper, the implemented design supports a wide set of with advanced hardware and networking functionalities such as store & forward transmission, error management, power and security. The capability to support all these above said features in the same hardware reduces the complexity overhead and the chip size will be drastically reduced. The proposed Network interface represents a complete solution which can be used in different scenarios from real time applications to multimedia broadcasting based on the system requirements by modifying some features to obtain an optimised hardware description and the results are obtained here on FPGA kit for real time applications slight modifications to be made as per the requirements.

REFERENCES

[1] Design of a NoC Interface Macrocell with Hardware Support of Advanced Networking Functionalities Saponara, Tony Bacchillone, Esa Petri, *Member, IEEE,* Luca Fanucci, *Member, IEEE,*Riccardo Locatelli, and Marcello Coppola

[2] NoC Advantages for SoC Prototyping on Big FPGA BoardsJonah ProbellArteris, Inc jonah@arteris.com

[3] P. S. Paolucci, F. LoCicero, A. Lonardo, M. Perra, D. Rossetti, C. Sidore, P. Vicini, M. Coppola, L. Raffo, G. Mereu, F. Palumbo, L. Fanucci, S. Saponara, and F. Vitullo, "Introduction to the tiled HW architecture of SHAPES," in *Proc. Design, Automation and Test in Europe*, 2007,pp. 77–82.

[4] B. A. A. Zitouni and R. Tourki, "Design and implementation of network interface compatible OCP for packet based NOC," in *Proc. 5th Int Design and Technology of Integrated Systems in Nanoscale Era (DTIS) Conf*, 2010, pp. 1–8.

[5] T. Tayachi and P.-Y. Martinez, "Integration of an STBus Type 3 protocol custom component into a HLS tool," in *Proc. 3rd Int. Conf. Design and Technology of Integrated Systems in Nanoscale Era DTIS 2008*, 2008, pp. 1–4.

[6] "NoC Interface for fault-tolerant Message- Passing communication on Multiprocessor SoC platform," in *Proc. NORCHIP*, 2009, pp. 1–6.
[7] "Synthesis of networks on chips for 3D systems on chips," in Proc. Asia and South Pacific Design Automation Conf. ASP-DAC 2009, 2009, pp. 242–247.
[8] "Efficient 2DMesh Network on Chip (NoC) considering GALS approach," in Proc. Fourth Int. Conf. Computer Sciences and Convergence Information Technology ICCIT '09, 2009, pp. 841–846.

Veeraprathap.V currently pursuing his Post graduate degree in Digital Electronics from Sri Siddhartha Institute of Technology, he received his Bachelor degree in Electronics and Communication Engineering from Visvesvaraya Technological University, Belgaum, Karnataka, India. His research interests are in the areas of Networking, CMOS VLSI and Wireless Communication.

Dr M Nagaraja working as an Associate Professor in the Department of Electronics and Communication, Sri Siddhartha Institute of Technology, Tumkur, Karnataka, INDIA. He has completed his Master's and Ph.D degree in the field of Electronics in the year of 2005 and 2010 respectively. He has published four research papers in reputed international journals published by Elsevier and IEEE. Presented research papers in eight international conferences held at different places across INDIA. His research interest is in the field of Nano Electronics, Networks, Wireless Communication and Microwave devices.

Dr.M.Z.Kurian received his Bachelor Degree from Bangalore University and Post graduate degree in Industrial Electronics from Mysore University, and Ph.D degree in Software Engineering from Dr.MGR University, Chennai, Tamil Nadu, India. He has more than 30 Years of Teaching in the field of Electronics & Communication Engineering. Published several papers in peer reviewed international journals including IEEE, and several conference papers.