ABSTRACT

Present day System-On-Chip (SoC) devices are made of hundreds of cores to be able to handle the complex functionality implemented in modern day devices. Intercommunication requirements for SoCs made of hundreds of cores will not be feasible using a single shared bus or a hierarchy of buses due to their poor scalability with system size and sharing of the bandwidth between all the cores. To overcome these problems of scalability and complexity, Network-on-Chip (NoC) has been proposed as a promising replacement for buses and dedicated interconnections.

Our project aims at the design and verification of a 2x2 Mesh Topology based NoC System in a multicore environment built using OpenRISC OR1200 processors. This report presents the project objectives; previous work and motivation for the project; design and verification methodology; performance evaluation methodology and the results obtained from RTL simulations.

Keywords


1. INTRODUCTION

With rapid developments in VLSI technology, integrated circuits have become smaller and smaller, and the number of modules integrated on a single chip have multiplied significantly. As a result, providing efficient interconnections among numerous on-chip cores and peripheral modules has become a primary concern in chip design. Traditional Buses can efficiently connect 3-10 communication elements but cannot scale to higher numbers. In addition, long, global wires have become undesirable for use due to their low performance, high power consumption and noise phenomenon. Recent research has shown that the traditional bus based architectures and point-to-point based interconnects have become a bottleneck with regard to speed-area-power performance metric [1].

A design paradigm called Network-on-Chip (NoC) has been proposed as a solution for interconnection of chip multiprocessors to transfer the maximum amount of information within the least amount of time (and within cost and power constraints) [2]. NoC consists of Network Interfaces, Routers, set of links interconnecting the Routers and a defined communication protocol for IP core interaction [14]. NoC offers significant advantages over the conventional bus schemes and point-to-point architectures for building complex and high performance SoC Architectures. The NoC based architecture differs from the bus based architectures in two major concepts. NoCs use packet based communication rather than a circuit switched network, as is the case in a Bus. NoCs have a decentralized and distributed network structure, rather than a global shared bus or a centralized architecture.

Our project aims at the design, implementation and verification of a 2x2 Mesh NoC System constructed using two Master OR1200 processor cores, single unified Memory core and a UART IP core. Network components like Network Interfaces and Routers have been designed, implemented and are interconnected to form a multicore NoC System. A complete functional verification environment has been built to assess the performance of the developed NoC system. Performance metrics like latency, throughput, bisection bandwidth, frequency of operation, area and power have been measured for the designed NoC System. To compare the performance metrics of the NoC system, a Bus Based communication system comprising of OR1200 cores, Memory and UART cores has been built. Finally, to test the maximum traffic supported by the designed NoC Network, a test environment comprising of Traffic Generators and Traffic Analyzers that simulates random traffic patterns has been developed.

Section 2 of the report mentions the related research in the field of NoC. Section 3 deals with the description of architecture and implementation details of the various network components constituting the 2x2 NoC System, bus based and test environments. Section 4 details the performance evaluation methodology. Section 5 discusses the results obtained from RTL simulations, Section 6 presents the conclusions we arrived at and section 7 describes the future work.

2. LITERATURE SURVEY

Significant amount of research has been done to evaluate the design and implementation aspects of NoC during the last few years. Several researchers have proposed various network topologies, switch architectures, routing algorithms and power efficient NoC designs. Many approaches have been implemented in FPGA for accurate functional validation of the NoC system. In [3], NoC concept for platform FPGA’s has been presented. NoCs with Mesh Topology have been implemented using a crossbar switch on a platform FPGA. Platform FPGA’s have been shown to offer programmability in terms of the NoC topologies that can be implemented, thus making it suitable for multiple applications. In [4, 5, 6], a complete HW-SW NoC emulation framework has been presented, that compares NoC features at physical level. This emulation framework is implemented on an FPGA platform. It
consists of Traffic Generators (TG), Traffic Receptors (TR), a Control module and the Network to be emulated. The Traffic Generators inject random traffic into the network and the Traffic Receptors verify the data integrity of packets received from the network. These modules are interconnected by a bus and the platform can support up to 1024 TG/TR’s. In [7], the Micro architecture of OpenSPARC T1 processor is described. The OpenSPARC processor consists of a three stage pipelined crossbar switch to establish communication between its eight processor cores, four banks of L2 Cache memory, I/O Devices and Floating Point Unit. Centralized Arbitration scheme is implemented with the oldest requestor getting the highest priority. In [8], a new Multi Processor SoC (MPSoC) is presented, which is used for studying different interconnection alternatives implemented in state of the art FPGA technology. A thorough analysis of performance and area tradeoffs for both bus and NoC based designs has been presented. In addition to these, our literature survey titled “Network on Chip Architectures for Real Chip Implementations” describes the implementation features of NoC architecture in several Industrial processors like Intel Teraflop, IBM Cell Processor, Tilera TILE64 and academic processors like UTAustin TRIPS, MIT RAW and KAIST BONE. Comparison of various NoC features of these processors has been presented in this literature survey [15].

Even though previous approaches validate several NoC implementation features, very few among these test interconnection mechanisms with real applications. Traditional communication approaches use bus architectures to establish communication between processor cores and memory. Our approach explores the possibility of connecting multiple cores and memories through an on-chip communication network, designed to provide high performance, low latency and low area overhead. This designed system can be expanded to accommodate more number of cores with minor changes in the current architecture, thus complying with the scalability requirements.

For validating the NoC switch, [3, 4, 5, 6] use traffic generators to generate traffic patterns. These may not model the real application traffic scenarios when several processor cores interact over a NoC. Our approach overcomes this limitation by using OpenRISC processors to generate actual traffic patterns. Most of the previous approaches use FPGA as a platform for validating the NoC architecture. Our approach will concentrate on the verification of the NoC using HDL simulations. Our work will aim at implementing network architecture similar to the OpenSPARC T1 architecture described in [7].

3. DESIGN
3.1 2x2 NoC System
A 2x2 NoC system has been designed to interconnect two OR1200 OpenRISC cores with Memory and UART using a 2D Mesh network. The Mesh network consists of four routers, with each router connected to an IP core and other routers. The block diagram of the designed NoC system is shown in Figure 1.

The NoC system comprises of OpenRISC OR1200 cores, Memory, UART, Network Interfaces and Routers. For validating the NoC switch, [3, 4, 5, 6] use traffic generators to generate traffic patterns. These may not model the real application traffic scenarios when several processor cores interact over a NoC. Our approach overcomes this limitation by using OpenRISC processors to generate actual traffic patterns. Most of the previous approaches use FPGA as a platform for validating the NoC architecture. Our approach will concentrate on the verification of the NoC using HDL simulations. Our work will aim at implementing network architecture similar to the OpenSPARC T1 architecture described in [7].

A brief description of each of the components is given below.

3.1.1 System Components
3.1.1.1 OR1200 OpenRISC Core
OR1200 is a 5-stage integer pipelined, 32-bit RISC machine with separate memory management units (MMUs) for Instruction and Data Memory. The instruction and data caches are 1-way direct-mapped with a capacity of 8KB and a line size of 16-byte. The OR1200 implements ORBIS32 Instruction Set Architecture which supports five instruction formats and two addressing modes: register indirect with displacement and PC-relative [9] [10].

3.1.1.2 UART and FLASH
The UART model used in this system is 16550 UART (Universal Asynchronous Receiver/Transmitter) and is used for serial communication. The Flash memory used in this system is Intel’s SmartVoltage 28F016S2 Flash with a capacity of 2MB.

3.1.1.3 WISHBONE Interface
All the above mentioned IP cores communicate with the external world using a common interface called WISHBONE Interface [11]. WISHBONE is a bus protocol standard used for interconnecting IP cores using various interconnection schemes like Point-to-Point, Data Flow, Shared Bus and Crossbar Switch. A WISHBONE bus connects Wishbone master (initiates bus transactions) and Wishbone slave (responds to request initiated by the master). In the designed NoC system, OR1200 cores act as Wishbone masters, and Memory and UART act as Wishbone slaves.

3.1.1.4 Network Interface (NI)
Network Interface connects the IP Cores with the main network. The NIs connected to the OR1200 IP Cores, termed as Slave NIs, convert the wishbone signals from the IP cores to packets whereas the NIs connected to Memory and UART, termed as Master NIs, convert the packets received from the network into Wishbone signals.
The packet format for the designed NoC is as shown in Figure 2.

<table>
<thead>
<tr>
<th>S A</th>
<th>D A</th>
<th>PKT NUM</th>
<th>WB ADDR</th>
<th>WB DATA</th>
<th>E N</th>
<th>R R</th>
<th>BYTE SEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>80</td>
<td>79</td>
<td>78</td>
<td>77</td>
<td>70</td>
<td>69</td>
<td>38</td>
</tr>
</tbody>
</table>

Figure 2: 82-bit wide Packet Format

Each IP Core is assigned a position in the coordinate system. The SA field of the packet contains the source address coordinates and the DA field contains packet’s destination address coordinates. PKT NUM field denotes packet number which is unique for each packet generated by NI. SA, DA and PKT NUM fields constitute the header of the packet. The WB ADDR, WB DATA, WEN, ERR and BYTE SEL are the wishbone signals and constitute the packet payload. WB ADDR field contains the address issued by the IP Core and the WB DATA field contains the data read/written from/to the slave IP Core. WEN is the write enable signal issued by the IP Core to indicate a write operation. ERR is the error signal generated in case of bad transactions like a packet reaching a wrong destination. BYTE SEL issued by the IP Core is the data qualifier for each byte of the WB DATA.

3.1.1.5 Router

The Router is one of the main building blocks of the NoC system. It consists of five ports - North (N), South (S), East (E), West (W) and the fifth port, connected directly to the Processing element which can be the OR1200 Processor Core, Memory or UART communication device. It is from the fifth port that packets enter or leave the network. In the current design, only three ports of each router are used. The other ports can be used to scale the design to a 4x4 or 8x8 2D mesh.

The router basically comprises of five main components which are listed below and shown in Figure 3

1. Input FIFOs
2. Arbiter
3. Decoding and Routing Block
4. Output FIFOs
5. Flow Control

Figure 3: Router Block Diagram

A detailed description of each of these blocks is given below:

3.1.1.5.1 Input FIFOs

The Input FIFOs are used to buffer packets received from various input ports of the router. There is one FIFO per input port of the router. Each FIFO has a width of 82 Bits (Packet Size) and a depth of 4 packet locations. In the current design, the limitations of Wishbone protocol cause at most one location to be occupied in the Input FIFOs. However, the depth has been fixed to 4 packets to account for the scalability of the design.

3.1.1.5.2 Arbiter

An Arbiter is used to select one among several input ports requesting access to an output port. The requestors (FIFOs) send a request signal and the arbiter responds with a grant to one of the several requestors [12]. Several arbitration schemes like Round-Robin Arbiter (RR), Weighted Round-Robin Arbiter (WRR) and Priority based are available to decide which requestor will be granted access [13]. The current design implements the Round-Robin Arbitration scheme, wherein all the requestors are serviced in a circular manner. In this scheme, all the requestors get a fair chance to access the output resource.

3.1.1.5.3 Decoding and Routing Block

The packet received from the Arbiter block is first decoded and then routed to its destination. The routing scheme implemented in our design is Dimension Ordered X-Y routing [13]. In this routing scheme, a packet is first routed along the X-direction and then along the Y-direction. In this scheme, the path of a packet is predictable. One of the main reasons for choosing X-Y routing lies in its ease of implementation in Mesh topology where each IP core is represented in x-y coordinates. Also, implementation of X-Y routings ensures deadlock avoidance by restricting the formation of resource dependence cycles.

3.1.1.5.4 Output FIFOs

The output FIFOs are used to buffer the packets processed by the decoding and routing logic block. Buffering of the packets is necessary to prevent packet loss in case of traffic congestion in the network. There is one FIFO per output port. Each FIFO has a width of 82 bits (Packet Size) and a depth of 4 packet locations. In the current design, due to the limitations of the Wishbone protocol each FIFO will have a maximum of two packets in the worst case. However, the depth of the output FIFOs has been fixed to 4 to account for the scalability of the design.

3.1.1.5.5 Flow Control

Flow Control is a mechanism which decides how network resources are allocated to packets transmitted over the Network. There are several flow control methods like Credit-Based Flow Control, On/Off Flow Control and Ack/Nack Flow Control [13]. The current design implements a Credit Based Flow Control mechanism. Each output port of the router has a Flow Control block except the port connected to the Processing Element. A Flow Control module implements a credit counter which indicates the occupancy of the Input FIFO of the adjacent router. This counter is decremented when a packet is transmitted to the adjacent router and is incremented when the packet is freed from the Input FIFO of the adjacent router.

3.1.2 Packet Flow

Our 2x2 NoC system implements Crossbar switch Wishbone scheme for connecting the two wishbone masters (OR1200 cores) to the two wishbone slaves (Memory and UART).

The Slave Network Interface (SNI), connected to the OR1200 cores, converts the OR1200’s Wishbone signals to packets, which are injected into the Network. Before injecting the packets into the
Network, the SNI generates a unique 8-bit message number for each packet and transmits it along with the data in the packet (PKT NUM field). This message number is also stored in the SNI and is compared with the message number received along with the ACK packet from the Slave Core. The SNI also decodes the incoming address from the OR1200 cores and assigns a destination port address (in terms of x-y coordinates) to the packet. Apart from the Message Number and Destination Address generation, the SNI also appends the Source Port’s Address (in x-y coordinates) for each packet. Once the packet is ready, SNI generates the Input FIFO Write Enable to the Processing Element’s (PE) Input FIFO. The packet is written into the Input FIFO, provided Input FIFO Full has not occurred. In case of Input FIFO Full condition, the packet will have to wait for Input FIFO Full signal to go low and will be buffered in the SNI. On the receiving end, the SNI converts the packets from the Network to WISHBONE signals, which are then forwarded to the Master OR1200 core. By comparing the Message Numbers in the transmitted and the received packets, any packet loss condition that would have occurred during the packet transit can be detected. On packet reception, the SNI asserts the ACK WISHBONE signal to the OR1200 core, once the message number of the received packet is validated. SNI asserts an ERR WISHBONE signal in case of erroneous transaction or message number mismatch.

Once the Packet is written into the router’s Input FIFO, the FIFO issues a REQUEST to the Arbiter and waits for its turn to be read out by the Arbiter. The Arbiter services the REQUEST and issues a GRANT to the requestor Input FIFO (in form of FIFO Read Enable) based on a round-robin selection scheme. The Arbiter then forwards the packet from the selected port to the Decoding and Routing Block.

Based on the packet’s Destination Address, current position of the switch/router (in terms of x-y co-ordinates) and the input port from which the packet has arrived, an output port is selected by the decoding/routing block and the packet is forwarded to the selected port’s Output FIFO. The Decoding and Routing Block generates an Output FIFO Write Enable signal for writing the packets into the selected Output Port FIFO.

Each output port in a Router is physically connected to either the next router’s input port or to the processing element port (via the NI). The Flow Control block generates the current router’s output FIFO Read Enable and also asserts the next router’s input FIFO Write Enable control signals. If the packet reaches the router connected to its destination, the Master Network Interface (MNI) issues a Read Enable to the Output FIFO of the Processing Element Port, if the Slave Core is ready to process the next incoming request. The MNI extracts the Message Number, Source Address and Destination Address fields from the Incoming Packet and stores them internally. The various fields in the packet are then decoded and forwarded to the Slave Core in the form of Wishbone signals. On reception of an ACK from the Slave core, the MNI attaches the stored Message Number extracted earlier to the Message Number field of the Acknowledge packet, which contains the data from Slave core. The stored Source Address coordinates are inserted into the Destination Address Field and stored Destination Address bits are inserted into the Source Address Field of the Acknowledge packet. This packet is then forwarded to the Master OR1200 core through the network in a similar manner. On reception of an Acknowledge packet, the OR1200 Master core ends the current transaction and initiates a new transaction.

3.2 Bus Based System
To compare the performance of the designed 2x2 NoC system, a bus based communication system is built with the OR1200 cores, Memory and UART modules as shown in Figure 4.

In this system, if any two IP cores want to communicate simultaneously with the other IP cores, they request access to the shared bus at the same time. The bus arbiter grants access to only one IP core at a time and the other IP core has to wait till the previous transaction is completed and the bus becomes free. This way, the transactions in a common shared bus system happen in a serialized manner.

3.3 Test Environment
In our designed 2x2 NoC system, OR1200 cores issue an instruction only if they receive an acknowledge corresponding to their previous requests. This is a limitation of the wishbone bus protocol and as a result, our network may not be fully utilized. To obtain the maximum throughput supported by our network, a test environment is built, where all the IP Cores are replaced by Random Traffic Generators and Traffic Analyzers. The test environment is shown in Figure 5.
In this environment, each Traffic Generator (TG) connected to a router generates random traffic patterns with each packet of the traffic pattern assigned a unique Message Number. The TG then adds the packet into the scoreboard, which is indexed using the Message Number of that packet. The Traffic Analyzers (TA) present at each router compare the packet received from the network with that available in the scoreboard information to ensure data integrity. With the random traffic generated at each router, the network is subjected to maximum utilization and the maximum throughput supported by the network is obtained.

4. PERFORMANCE EVALUATION METHODOLOGY

For performance evaluation, three environments were considered. The first environment is the 2x2 NoC system explained in section 3.1. To establish communication of OR1200 cores with Unified Memory and UART, two independent programs are run, one on each core. The second environment is the bus based system explained in section 3.2. The bus based system connects the two OR1200 cores, Unified Memory and UART in the same configuration as the 2x2 2D-Mesh NoC system. The programs used to evaluate the performance of the three systems are listed in the Table 1.

Table 1: Programs run in OR1200 cores

<table>
<thead>
<tr>
<th>CONFIG</th>
<th>2x2 2D MESH</th>
<th>BUS BASED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C1- BASIC</td>
<td>C1- BASIC</td>
</tr>
<tr>
<td></td>
<td>C2-BASIC</td>
<td>C2-BASIC</td>
</tr>
<tr>
<td>2</td>
<td>C1- BASIC</td>
<td>C1- BASIC</td>
</tr>
<tr>
<td></td>
<td>C2-UART</td>
<td>C2-UART</td>
</tr>
<tr>
<td>3</td>
<td>C1- DHRY</td>
<td>C1- DHRY</td>
</tr>
<tr>
<td></td>
<td>C2-UART</td>
<td>C2-UART</td>
</tr>
</tbody>
</table>

BASIC program tests the basic instruction set of the OpenRISC ISA by exercising all the basic instructions. UART program performs a UART access by configuring the UART registers and performs basic serial communication using the UART communication device. DHRY is the standard Dhrystone benchmark. Dhrystone is a synthetic integer benchmark containing a main loop executed number of times [16].

4.1 Performance Metrics

The following parameters were considered for performance evaluation.

1. **Processor Centric Latency** – This is defined as the sum of the time required for the request issued by the processor to travel all the way to the slave IP cores (Memory or UART) and the time required for the acknowledge issued by the slave IP cores to travel back to the master processor cores.

2. **System Throughput** – The system throughput is defined as the sum of rates at which packets arrive at all destinations from all the sources in a defined system.

3. **Bisection Bandwidth** – This is defined as the bandwidth between two parts of a network if the network is segmented into two equal parts [17].

4. **Router Latency** – This is defined as the number of clock cycles required by the router to process a packet.

5. **Router Throughput** – This is defined as the sum of the output packet rates on all the ports of a router.

4.2 Measurement of Parameters

4.2.1 Processor Centric Latency

For measuring Processor Centric Latencies, Latency Monitors were built around each OR1200 core to monitor the wishbone bus. The Latency Monitor keeps track of the period between the triggering of a new Wishbone transaction (as indicated by the Wishbone Strobe Signal) by the OR1200 core and reception of an Acknowledge from the slave IP core for transaction initiated by the OR1200 core. This period gives the processor centric latency with respect to each OR1200 core.

4.2.2 System Throughput

For measuring the System Throughput, Throughput Monitors were built around each slave IP core. Throughput was measured at each of these slave IP cores for each source-destination pair. At each destination slave IP core, a packet counter keeps track of the number of packets received from a particular source.

Throughput was calculated using:

Throughput = (Number of Packets received at a fixed destination from a particular source / Total number of clock cycles) * clock frequency (in MHz) * data bits transmitted in a packet

The Total System Throughput is finally calculated by summation of throughputs obtained for all destinations with respect to all sources.

4.2.3 Router Throughput

To measure Router Throughput, a Router Throughput Monitor was built as a wrapper around each router. Router Throughput was calculated by summing up the packet rates on all the output ports of a router.

Throughput for each output port was measured using:

Throughput (w.r.t each output port) in Mbps = (Number of Packets received at a particular output port of a router/ Total number of clock cycles) * clock frequency (in MHz) * data bits transmitted in a packet.

4.2.4 Bisection Bandwidth

The Bisection Bandwidth was measured across a cross-sectional plane of the network by summing up the packet rates on output ports of two adjacent routers connected in a horizontal plane.

A comparison was made between the 2x2 NoC system and the bus based system with respect to all of the above mentioned parameters. To obtain the maximum throughput supported by a router, the test environment described in section 3.3 was considered. The Router Throughput was calculated in a manner similar to the one explained above.
4.3 Synthesis
The designed 2x2 NoC network was synthesized using Synopsys Design Compiler using 180nm Standard Cell Technology library. Parameters like Router Area, Dynamic power and Leakage power consumed by the router were obtained from the synthesis results. Also, the design was closed for timing and the maximum frequency supported by the network was obtained.

5. RESULTS
5.1 2x2 NoC System v/s Bus Based System
Performance metrics like System Throughput and Processor Centric Latency measured for the designed 2x2 NoC system were compared with those obtained for the traditional Bus Based system running under the same configuration. The worst case latency results for the two processor cores C1 and C2, obtained from the simulations, are indicated in Figure 6 and Figure 7 shown below.

5.1.1 Observation I
Figure 6 and Figure 7 clearly indicate an improvement of latency in a NoC based environment over bus based environment for all the three test cases that were simulated. In a bus based system, worst case latency for a core occurs when the bus is busy servicing another transaction. In this case, request issued by a core has to wait till the bus becomes free and hence the waiting time adds on to the latency. In a NoC system, worst case latency occurs when both the processor cores access the same slave simultaneously. In this case, the request is routed through the network and the packet waits in the Output FIFO of the destination router. The packet is processed once the slave finishes processing the other core’s request. Hence, the latency overhead is just the time taken by the slave IP core to process a request.

5.1.2 Observation II
It is seen from the charts that the worst case latency for Core2 is higher than the worst case latency for Core1 in the bus based environment. This can be explained as follows.

5.1.3 Observation III
Configuration 1 (C1-Basic and C2-Basic) gives the highest latency values for the bus based environment. This is due to the fact that in this configuration both the processor cores access only memory and there are no UART accesses. UART responds faster to requests as compared to Memory. Therefore, when both the cores access memory, they have to wait for a longer time before their requests are serviced taking into account the memory access time. This is not the case in the other two configurations which have intermediate UART accesses. In the NoC based environment, the worst case latency is almost constant across all the test cases because the requests to the same slave from both the cores can be launched simultaneously.

5.1.4 Observation IV
Figure 8 shows the throughput comparison for the Bus based and the NoC based systems. It is clearly seen that the NoC based system offers a much higher Throughput (almost double) as compared to the Bus based environment. This is due to the fact that in NoC based environments, multiple requests to the same slaves can be launched in parallel. In a Bus based environment, the requests are launched in a sequential fashion and each request has to wait for the previous request to be completed. This results in a lower throughput in a bus based system.

In the Bus based environment, the bus arbiter assigns a lower priority to Core2’s requests when compared to Core1’s requests. As a result, if there are more requests from Core1 to the slave cores, Core2 will have to wait for a longer period before its request can be serviced. The latency difference is clearly seen in the first configuration (C1-Basic and C2-Basic) where both the cores request access to memory simultaneously and Core1’s requests are given priority over Core2’s requests. This difference in the Core1’s and Core2’s worst case latency is reduced in the NoC based environment because in the NoC based environment, the requests from both the cores can be launched simultaneously and will not have to wait for the other core’s request to be completed.
5.2 Network Characteristics

Table 2 summarizes the characteristics of the designed 2x2 2D-Mesh Network.

<table>
<thead>
<tr>
<th>CHARACTERISTIC</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology</td>
<td>2x2 2D-Mesh</td>
</tr>
<tr>
<td>Network Operating Frequency</td>
<td>150 MHz</td>
</tr>
<tr>
<td>Packet Width</td>
<td>82 bits</td>
</tr>
<tr>
<td>Input &amp; Output FIFO buffer depth</td>
<td>4</td>
</tr>
<tr>
<td>Arbitration Scheme</td>
<td>Round Robin</td>
</tr>
<tr>
<td>Routing Algorithm</td>
<td>X-Y dimension order</td>
</tr>
<tr>
<td>Flow Control Scheme</td>
<td>Credit Based</td>
</tr>
<tr>
<td>Bisection Bandwidth</td>
<td>242 Mbps</td>
</tr>
<tr>
<td>Network Area</td>
<td>2081900</td>
</tr>
<tr>
<td>Network Dynamic Power</td>
<td>161.35 mW</td>
</tr>
<tr>
<td>Network Leakage Power</td>
<td>3.5 uW</td>
</tr>
</tbody>
</table>

The designed 2x2 2D-Mesh can be easily scaled to a 4x4, 8x8 or a 16x16 Mesh Network. The packet size is fixed to 82 bits and includes wishbone address and data signals. Round-Robin arbitration scheme was used for input packet arbitration and Dimension Ordered X-Y Routing scheme was used as the routing algorithm. Credit based Flow Control scheme was implemented in the design. The Maximum Operating Frequency, Network Area, Network Dynamic Power and Network Leakage Power were obtained from synthesis using 180nm Standard Cell Library.

5.3 Router Characteristics

The characterization of the designed router was done in both the 2x2 2D Mesh system as well as the Test Environment. The characteristics of the router are indicated in the Table 3.

<table>
<thead>
<tr>
<th>CHARACTERISTIC</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ports Per Router</td>
<td>5</td>
</tr>
<tr>
<td>Router Frequency</td>
<td>150 MHz</td>
</tr>
<tr>
<td>Router Latency</td>
<td>9 Cycles</td>
</tr>
<tr>
<td>Throughput</td>
<td>233.28 Mbps (2x2 NoC System)</td>
</tr>
<tr>
<td>Throughput</td>
<td>4.89 Gbps (Test Environment)</td>
</tr>
<tr>
<td>Area</td>
<td>900646</td>
</tr>
<tr>
<td>Dynamic Power</td>
<td>84 mW</td>
</tr>
<tr>
<td>Leakage Power</td>
<td>1.5 uW</td>
</tr>
</tbody>
</table>

The Test Environment built using Traffic Generators and Traffic Analyzers enabled us to exercise maximum traffic conditions and measure the network’s maximum throughput and frequency of operation supported. The NoC system could be easily scaled to accommodate more processor cores, thereby increasing the scalability limit. Because of all the above mentioned advantages over buses, NoC has evolved as an emerging paradigm for interconnecting various IP cores.

6. CONCLUSIONS

This project aims at introducing the concept of Network-on-Chip and demonstrates that NoC based architectures achieve better performance in terms of parameters like latency, and throughput when compared to the conventional bus based systems. With our designed NoC system, multiple processor cores are shown to gain access to the network simultaneously. This is not true in case of bus based schemes, where accesses to the bus are serialized. The Bus Based Environment enabled us to directly compare the performance results obtained with that of our NoC based system. It is clear from the results that NoC systems achieve lesser worst case latencies and higher throughput as compared to the Bus based systems. The difference in latencies between NoC and Bus based architectures will become significant with increase in the number of processor cores requesting access simultaneously.

The NoC system was synthesized using Synopsys Design Compiler with 180nm Standard Cell Library. The NoC based system can be easily ported into an FPGA. Maximum Frequency supported by the synthesized network was found out to be 150MHz and the network consumed an area equivalent to 2 million equivalent NAND gates. The dynamic power dissipated by the network was about 160mW. The 2x2 NoC system was able to achieve a system throughput of nearly 120 Mbps. Each router supported a maximum throughput of 234 Mbps. For the Test Environment, each router achieved a maximum attainable throughput of 4.89 Gbps, exhibited a latency of 9 cycles, consumed an area equivalent to 0.9 million equivalent NAND gates and dissipated dynamic power of 84mW.
7. FUTURE WORK
The current design implemented a Round-Robin (RR) arbitration scheme and X-Y routing algorithm. Arbitration algorithms possessing better fairness and efficiency can be designed and implemented. More efficient routing algorithms can be implemented in order to improve routing latency and deadlock avoidance. Optimizations can be done to improve the power, area consumption and latency of the designed NoC.

The designed 2x2 2D Mesh Network can be scaled to a 4x4, 8x8 or 16x16 Mesh to incorporate more processor cores and performance evaluation can be done. Finally, different network topologies can be implemented, evaluated and compared against the 2x2 2D mesh network.

8. ACKNOWLEDGMENTS
Our sincere thanks to Dr. Paul Gratz for his constant support and guidance throughout this project.

9. REFERENCES