Design & Simulation Of SMITHA: A Structured and Scalable Architecture For 3D Network on Chip Systems
Sanju V\textsuperscript{1}, Niranjan Chiplunkar\textsuperscript{2}, M Khalid\textsuperscript{3} and P Venkata Krishna\textsuperscript{3}
\textsuperscript{1}Nitte Meenakshi Institute Of Technology, Yelahanka, Bangalore, India
\textsuperscript{2}NMAMIT, Nitte P. O, Karkala, India
\textsuperscript{3}VIT University, Vellore, India

Abstract
Network on chip is a new paradigm in ASIC design, and new topologies are emerging from this domain of work, Scalable Modular Interconnect for Three dimensional High Performance Applications is a typical 3D architecture (SMITHA) exactly. In this paper, SMITHA will be analyzed, and be designed in Verilog HDL.

Keywords SMITHA, Network On Chip, Design and simulation

1 Introduction
Early VLSI designs were implemented using common bus architecture. As time passed by the density of the IC’s rose dramatically. Bus architecture failed to give the required amount of performance. This gave rise to a new architecture called network on chip\textsuperscript{[1-3]}. The idea behind this is to use concepts of data networking on silicon. However, network on chip is becoming the backbone of all high performance chips, many topologies have been introduced in this novel field. Mesh and torus\textsuperscript{[2,4]} being the most commonly used topology. In this paper we like to introduce SMITHA, a new three dimensional topology for network on chip based systems.

In this paper, a new three dimensional topology for network on chip based systems is introduced, and then, the internals of the topology and the implementation of the topology are explained.

2 A new 3D topology : SMITHA
SMITHA: Scalable Modular Interconnect for Three dimensional High performance Application (SMITHA) is a new three dimension topology for network on chip based systems. The construction of the topology can be explained as follows. The base topology is obtained by removing the root node and interconnecting the nodes of a complete binary tree along the same layer as shown in Fig.1. The nodes are addressed based on the layer in which they belong and relative position within the layer.

The three dimension topology is obtained by placing the above discussed base topology one over the other and by interconnecting the adjacent levels as follows.
Fig. 1 SMITHA Base Topology With 3 Layers

Fig. 2 SMITHA Base Topology With Three Levels Having Three Layers / Level
• The connections between odd level and an even are done by connecting the odd layers though the left and even layers through the right. For example interconnection between level one and two.
• Similarly the connections between even level and an odd are done by connecting the odd layers though the right and even layers through the left. For example interconnection between level two and three.

This is depicted in Fig.2.

3 Design and simulation of SMITHA

The architecture discussed above was designed and implemented using Verilog HDL. The code was tested and verified using ModelSim and synthesised for cyclone II FPGA on Altera DE2 - 70 board.

The design starts from the smallest unit in the topology which is the node which interconnect to form the topology. The node is designed with five interfaces which connect to the external world. Fig.3 gives a representation of a node.

Fig.3 Internals of A SMITHA Node

Fig.4 Internals of A SMITHA Node Interface
Each node has five interfaces which are named left interface (LI), right interface (RI), top left interface (TLI), top right interface (TRI), bottom interface (BI). These connect to the neighbouring node to do their specified operation. Fig. 4 shows the internals of each of the interface.

It consists of the connecting pins namely REQ, ACK, DATA, CLK which are request pin, acknowledgement pin, data pin and the clock pin. It consists of the routing algorithm in the routing logic and the whole control of the node in control logic. It also consists of temporary storage purpose namely the registers – send and receive and the buffers – send and receive.

3.1 Packet Format

The packet has three components namely the destination address, source address and the data bits.

```
<table>
<thead>
<tr>
<th>Destination Address</th>
<th>Source Address</th>
<th>Data Bits</th>
</tr>
</thead>
</table>
```

Fig. 5 depicts the same. The source and destination address consists of the level number to which the node belongs, the ring in the level and the position of the node in that ring. The level numbering starts from 0 to n - 1 where n is the largest number of level the architecture holds. Similar arguments hold good for ring number and node number which are within each level. Similar argument hold good for source addressing also. Data bits can be of any size depending on the problem. For example a node addressing (1, 0, 0, 2, 0, 0, 25) represents a data 25 from level two ring zero node zero to level one ring zero node zero. The total size of the packet is $2 \times (\log_2(n) + \log_2(m) + \sum_{k=1}^{m} 2^k) + t$ data bits.

3.2 Communication between nodes

Fig. 6 shows how to nodes in SMITHA topology communicates.

The nodes in SMITHA communicate with each other with first raising the REQ pin. The receiving node checks whether the receive buffer is full or not. If the receive is not full then busy bit and the ACK bit is set. The code illustrates the same.

```
Pueduo code on recieve_REQ  // Pseudo code when request signal is received
{
    If(!full(recievebuffer))    // Checking receive buffer is full or not
```
Once receiving the request signal, the data sending module will clock and send the data one bit at a time. After sending the data the busy bit of the sending module is kept low. On the receiving side, the data is received on with the help of receive register. After the reception of data, the busy pin is kept low. Now the packet is checked by the routing algorithm to decide the next path. If the send buffer of the next interface is not full then the packet is send to the next interface else it is kept in the receive buffer of the current interface. The pseudo code below illustrates the same.

```plaintext
Pseudo_code_onACK_sendingside
{
    for(i = 0 ; i < length(packet); i++)
    {
        CLK = 1;
        DATA = sendregister[i];
        Clk = 0;
    }
    busybit = 0;
}
```

**Fig.6 Two Nodes Communicate In SMITHA**
Pseudo_code_onACK_recieveside
{
    for(i = 0 ; i < length(packet); i++)
    {
        recieveregister[i] = DATA;
    }
    busybit = 0;
    nextinterface = route(recieveregister);
    if(!full(sendbuffer(nextinterface)))
    {
        store(nextinterfacebuffer);
    }
    else
    {
        store(recievebuffercurrentinterface);
    }
}

3.3 Control Logic
It is a integral part of the system. It basically checks and starts the operation of
sending the packet. Furthermore the packets which in the receive buffer are for-
warded to their respective location by the logic. The pseudo code below depicts
the same.

pseudocode_contrologic
{
    if(!empty(sendbuffer)&&!busybit)
    {
        busybit = 1;
        REQ = 1;
    }
    if(!empty(receivebuffer))
    {
        nextinterface = route_alg(packet);
        if(!full(sendbuffer(nextinterface)))
        enqueue(packet)
    }
}

3.4 Output verification
The above circuit was coded in verilog and tested using Modelsim and was down-
loaded to DE2 – 115 board to test the code. The code was tested for absolute
Fig. 7 Output Waveform
accuracy with different conditions. One of such output is as below (Fig.7)- It describes the packet going from (1,2,1) to (1,2,0) and then it moves to the second level to (2,2,0) and ends at (2,2,1).

4 Conclusion

The paper briefly draws a light into the new three dimensional architecture and discusses the FPGA implementation using verilog HDL by means of Altera’s Quartus and Modelsim for simulation purpose. The paper starts with the node implementation to the topology. The implementation discusses the internals of the node with every part of the node in detail. It clearly brings out the working of the module by the waveform shown.

References


**Corresponding Author**
Sanju V can be contacted at:sanjuv21@gmail.com.