International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 4, No.1 (Jan-2015)

http://dx.doi.org/10.12785/ijcds/040107

Design And Implementation of RiCoBiT: A Structured and Scalable Architecture For Network On Chip Based Systems

Sanju V[1], Niranjan Chiplunkar[2], Suresh Mandia[3], Nancy Jain[4], Nikita Jaiswal[5], M Khalid[6]

[1] Asso. Prof, Department Of Computer Science & Engineering ,Nitte Meenakshi Institute Of Technology, Bangalore, India
[2] Principal ,NMAMIT, Karkala, India
[3]PG Student, Warsaw *University of Technology , Warsaw*
[4]IT Engineer, CISCO Systems, Bangalore, India
[5]Analyst, Mckinsey & Company, India
[6]School Of Computing Science & Engineering, VIT University, Vellore, India

E-mail address: sanjuv21@gmail.com, niranjanchiplunkar@rediffmail.com, sureshmandia@yahoo.in, nancyjain2307@gmail.com, nikitajaiswal17@gmail.com, mkhalidbs@yahoo.com

Received 30 Jun. 2014, Revised 13 Nov. 2014, Accepted 20 Nov. 2014, Published 1Jan. 2015

Abstract: We are already in an era of a billion transistors on a silicon die. This was possible due to research advancement of silicon physics and process technology which enabled implementation of large, high performance, complex functionality. The backbone of any high performance computing system is the underlying interconnection network. Today's, high performance systems are designed using network on chip. The design was realised by placing the different processing modules using various topologies like 2D mesh, torus. These existing interconnects limit itself in terms of performance and scalability. This paper discusses a new structured and scalable topology - RiCoBiT : Ring Connected Binary Tree in brief as an alternative along with its design and HDL implementation.

Keywords: Network on Chip, Topology, Design, Implementation

1. Introduction

Semiconductor physics and process technology has advanced many folds from its inception. Starting with design with LSI (Large Scale Integration) scale with a few hundreds of transistors to ULSI (Ultra Large Scale Integration) scale embedded more than a billion transistors on a silicon die. This was supported by the advancement in lithography process also. The initial design was small in terms of size, complexity and was implemented using simple common bus architecture. The communication between the different modules was done using polling and arbitration techniques. As design size and complexity increased, several advancement and standards in communication and bus architecture was introduced. This includes multi bus structures, parallel buses, ring structures, master slave configuration, etc. These design and communication techniques prevailed and sustained for the last three decades. As the complexity increased, these techniques posed a serious limitation in terms of performance and scalability. The limitations in terms of performance and scalability were overcome using a new paradigm in ASIC (Application Specific Integrated Circuit) design called Network on Chip (NoC) [1, 4, 5, 6]. The paradigm introduced concepts of communication network on silicon die. In this the processing modules are arranged using different topologies like 2D mesh, torus [3] being the popular ones. The modules interact with each other by sending packets between the modules. The limitation of common bus architecture was largely overcome by the concept of Network on Chip using mesh and torus. But the growth of fabrication process enabled us to design very high density ultra VLSI scale applications. This growth in density poses a serious limitation in terms of performance

Email: sanjuv21@gmail.com, niranjanchiplunkar@rediffmail.com, sureshmandia@yahoo.in, nancyjain2307@gmail.com, nikitajaiswal17@gmail.com, mkhalidbs@yahoo.com and scalability for application designed using mesh and torus. This paper discusses RiCoBiT (Ring Connected Binary Tree) : a new structured and scalable architecture for Network on Chip based systems. This paper also studies the properties of the architecture, proposes a design for the same and implements the design using verilog HDL.

2. RICOBIT ARCHITECTURE

In Figure 1 given below depicts the RiCoBiT architecture with K + 1 rings. As depicted in the figure, the topology is evolved by interconnecting concentric rings. The nodes in the same ring are connected to form a ring. The nodes in the adjacent rings are connected using the relation 2n and 2n + 1 where n represents the node number in the lower ring. For example node 0 in level 1will be connected to 0 & 1 in level 2. The nodes are addressed relative to the ring they belong and position within the ring. For example node addressed as (2,0) will represent a node in level 2 and node position 0. The level numbering starts from one and node addresses starts from zero. The inner most ring will be addressed as ring one. The number of nodes in each ring is 2^L where L is the ring number. Also the total number of nodes with L rings is $\Sigma\ 2^L$.

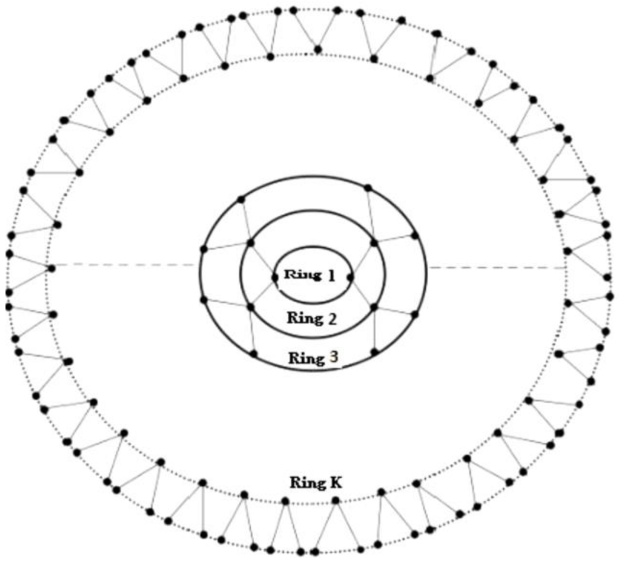


Figure 1. RiCoBit Architecture With K + 1 Rings

The architecture has numerous advantageous properties. The architecture is simple, regular and symmetric in nature. The architecture is very structured, modular and is scalable. The scalability property is well complemented with performance without significant increase in area. The architecture is well supported with a optimal routing algorithm. The existence of multiple shortest paths also makes the architecture very strong for adaptive routing in varying traffic conditions. It is observed that the architecture is superior to the currently existing popular architecture namely mesh and torus. The following sections discuss design and implementation of RiCoBiT.

3. DESIGN & IMPLEMENTATION

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 boards.

A. RiCoBiT Node Internals

A node is the basic building block of the topology. It is evident from the topology that a node can atmost connect to five neighbouring nodes. This is being depicting in figure 2 which presents the internals of a node.

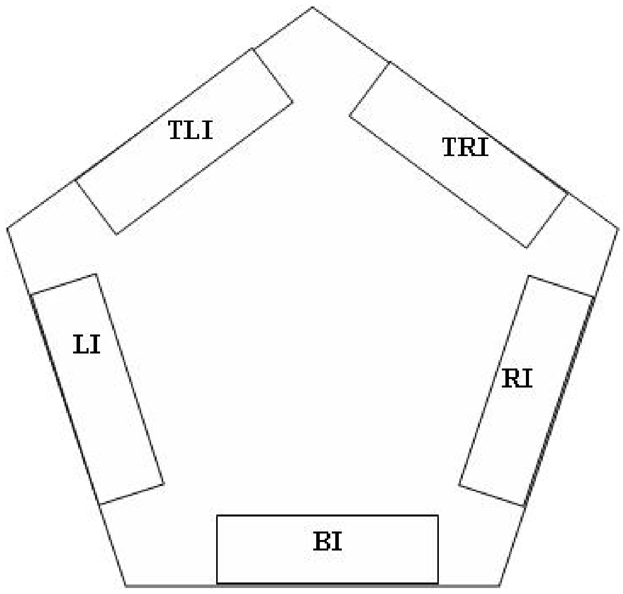


Figure 2. Internals of A RiCoBiT Node

It contains five interfaces namely bottom (BI), top left (TLI), top right (TRI), right (RI) and left (LI). Each of the interface has four bidirectional pins namely request (REQ), acknowledgment (ACK), data (DATA), clock (CLK). The interface also holds a temporary receive register for holding the data on reception and a temporary send register assisting in transmission of data. These registers are of the same size of the packet. In addition to the temporary registers, it also contains a send buffer which holds the packets that have to be sent through the interface and a receive buffer which holds the data in event of non acceptance at the next stage. The buffers are implemented using circular queue with each location as big as the size of the packet. The routing logic checks the destination address in the packet and determines the next node along its destination. The control logic then places the packet in the respective buffers. The busy bit indicates whether the interface is currently busy in data transmission / reception, this is depicted in figure 3.

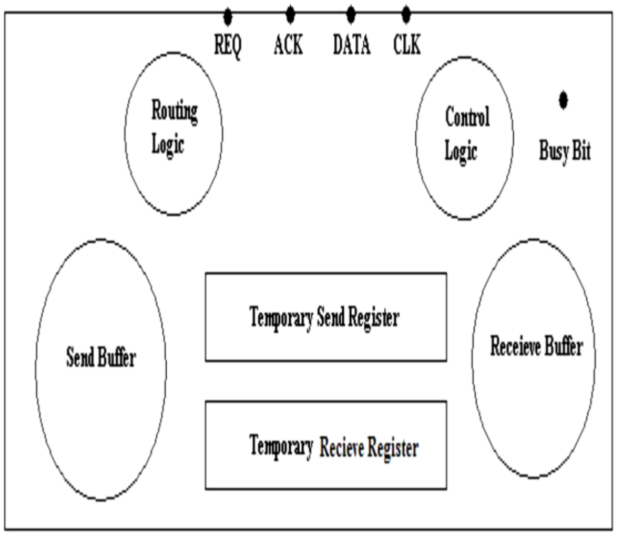


Figure 3. Internals of A RiCoBiT Node Interface

B. PACKET FORMAT & COMMUNICATION

The working of the node can be explained as below. The system is realised by placing the application process at the nodes. These nodes communicates with each other by exchanging packets. The format of the packets is as shown in Figure 4

Destination Address		Source Address		Data Bits
Ring No	Node No	Ring No	Node No	

Figure 4. RiCoBiT Packet Format

The packet has three fields namely the source address, destination address and data. The source address corresponds to the address where the packet is generated and destination address is where is packet is intended to be delivered. The source and destination field has two sub field which corresponds to the ring number where the node is located and the node number which is the position of the node within the ring. The packet will occupy $\log_2 R + R$ bits for source and destination field and a K bits data field where R is the maximum number of the rings in the configuration. So the total size of the packet is $2*(\log_2 R + R) + K$ bits. The modules interact with each other using a simple request acknowledgment protocol as shown in the Figure 5.

The process of communication starts when the data is in the send buffer and upon generation of a request signal from an interface of one node to the interface of the adjacent node. The busy bit is also set at this instance of

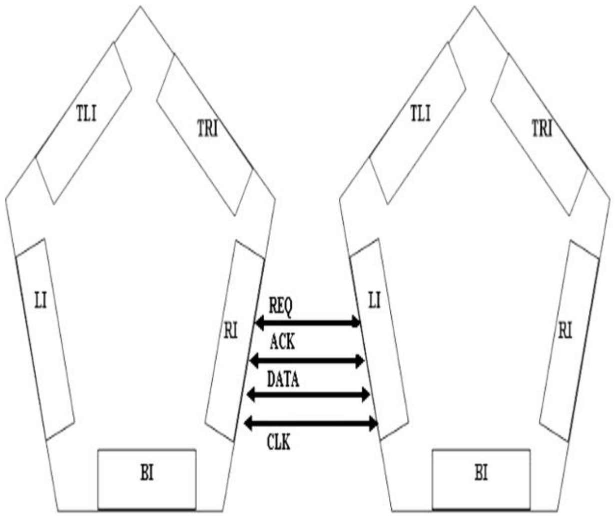


Figure 5. Interfacing Of Two Communicating RiCoBiT Nodes

time. On reception of the request signal the receiving node checks whether the receive buffer of the interface is full. If it is not full then an acknowledgment signal is returned for sending the data and sets its busy bit high. On reception of the acknowledgment signal, the sending interface generates a clock and the data is transferred serially from the send register via the data pin of the sending interface to the receive register of the receiving interface via the data pin. After the transmission of data, the busy bits of the interfaces are set low. The packet is then checked by the routing logic of the receiving interface and calculates the next node towards the destination. Now control logic checks whether the send buffer of the next interface is full. If not the packet is in queued in the send buffer else stored in the receive buffer of the receiving interface. The pseudo code in figure 6 illustrates the working of a similar example.

C. CONTROL LOGIC

The control logic is an integral part of the interface. After the reception of the packet in the receive register, the control logic with the help of routing logic checks the packet and calculates the next node towards the destination. Then it checks the send buffer of the connecting interface of the next node. If there is a vacant space then the packet in queued else it is store in the receive buffer. The control logic also has addition functionality of checking the send buffer and generates the request signal for sending the data as discussed above. Also the data in the receive buffer is transferred to the corresponding interfaces by this logic. This process continues till the buffers are empty. The pseudo code in figure 7 explains the same sequence.

```
pseduo_code_onREQ_recieve    //Request received at the interface
{
        If(!full(recievebuffer))
        {
                busybit = 1;
                ACK = 1;
        }
}
```

```
pseudo_code_on_ACK
@sendinterface
{
for(i=0;i<length(packet); i++)
{
        Clk = 1;
        DATA=
sendregister[i];
        Clk = 0;
}
      busybit = 0;
}
```

```
pseudo_code_on_ACK@
recieveinterface
{
for(i=0;i<length(packet); i++)

 {
receieveregister[i]=  DATA
}
 busybit = 0;
nextinterface=
route_alg(recieveregister);


if(!full(sendbuffer(nextinterface)))

enqueue_at_sendbuffer(nextinterface);

else

enqueue_at_recievebuffer(currentinterface);
}
```

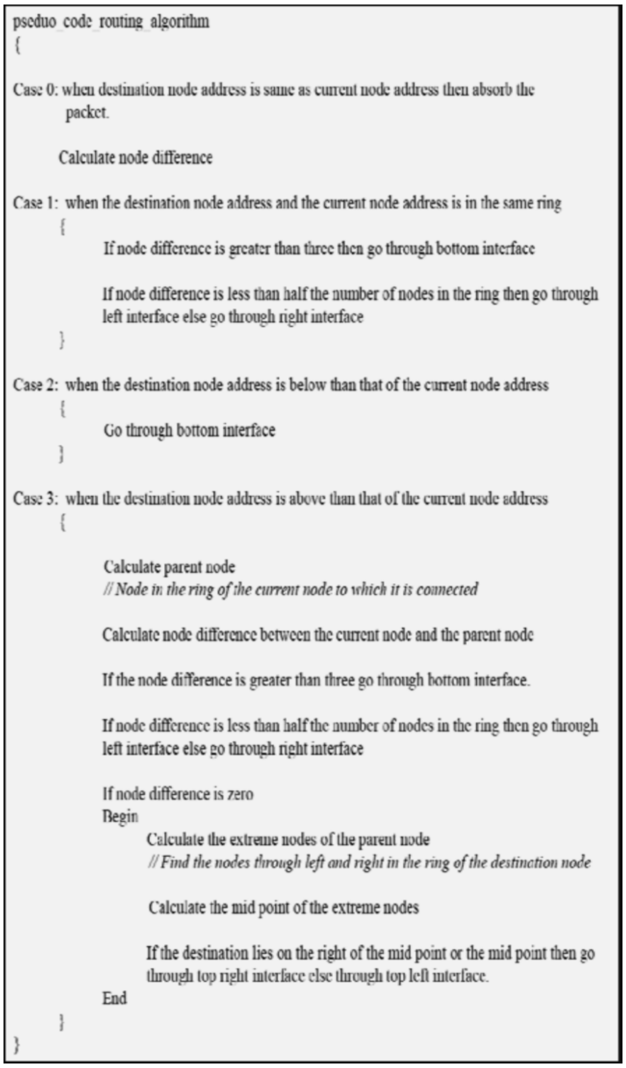
Figure 6. Pseudo Code Depicting The Working Of A Node

```
pseduo_code_controllogic
{
        if(!empty(sendbuffer) && !busybit)
        {
                busybit = 1
                REQ = 1
        }
        if(!empty(recievebuffer))
        {
                nextinterface = route_alg(packet);
                if(!full(sendbuffer(nextinterface))
                        enqueue(packet)
        }
}
```

Figure 7. Pseudo Code Depicting The Control Logic

D. ROUTING LOGIC

This section discusses an optimal routing algorithm. It routes the packet from the source to the destination through the shortest route with minimum number of hops. The logic checks the current node address and the destination address in the packet and calculates the next node along the destination. The routing algorithm is presented as a pseduo code in Figure 8. The routing algorithm was computationally verified and tested by comparing with shortest path algorithm (Floyds's). It is observed that the routing algorithm routes the packet from source to destination through the shortest path. i.e. the routing algorithm is optimal.



```
pseduo code routing algorithm
{

Case 0: when destination node address is same as current node address then absorb the
        packet.

      Calculate node difference

Case 1: when the destination node address and the current node address is in the same ring
      {
            If node difference is greater than three then go through bottom interface

            If node difference is less than half the number of nodes in the ring then go through
            left interface else go through right interface
      }

Case 2: when the destination node address is below than that of the current node address
      {
            Go through bottom interface
      }

Case 3: when the destination node address is above than that of the current node address
      {

            Calculate parent node
            // Node in the ring of the current node to which it is connected

            Calculate node difference between the current node and the parent node

            If the node difference is greater than three go through bottom interface.

            If node difference is less than half the number of nodes in the ring then go through
            left interface else go through right interface

            If node difference is zero
            Begin
                  Calculate the extreme nodes of the parent node
                  // Find the nodes through left and right in the ring of the destinction node

                  Calculate the mid point of the extreme nodes

                  If the destination lies on the right of the mid point or the mid point then go
                  through top right interface else through top left interface.
            End
      }
}
```

Figure 8. Pseudo Code Depicting RiCoBiT Routing Logic

E. DEADLOCK PREVENTION & PACKET LOSSES

The interface internals is designed in such a way that it prevents deadlock. This is done by giving fair chances for all the interfaces in the topology to transfer packets without starvation. This is realized using a choke bit in each of the interface. If an interface gets a chance of sending the data and the adjacent node has a packet to be transferred then, the adjacent node raises the choke bit which is connected to the adjacent interface. After the transmission is over, the interface would check the choke bit and if it is high, it allows the adjacent node to raise the request and transfer the data in the next cycle. This will ensure that the packet transfer happens alternatively between every pair of connected interfaces preventing deadlock.

The communication protocol is designed in such a way that the acknowledgment is raised only if there is a free space in the receive buffers. Also the presence of busy bit will ensure that request is not raised whenever the interface in involved in data transmission and reception. These mechanisms will ensure that the packet is safe when there are in the buffers or registers.

4. VERIFICATION & SIMULATION OUTPUTS

A node which is the building block of the topology was implemented using verilog HDL. The code was verified for functionality using ModelSim. The code was synthesized for cyclone II FPGA on DE2- 70 board using Altera Quartus. The wave form (Figure 9) depicts different stages that a packet takes after reaching the node's receive register till its exits. It also depicts the different stages of communication between two nodes. The different stages shown in the wave form are as below

1) Packet is in the receive buffer on the left interface of node 1
2) The routing algorithm is performed on the packet and put in the right interfaces send buffer.
3) Packet come to the send register node 1
4) Node 1 raises request signal to node 2.
5) Node 2 acknowledges the request.
6) Serial transfer of the packet from the send register of node 1 to the receive register of the node2.
7) Routing algorithm is performed and cycles continue.

The node is tested for simultaneous working of all the interfaces. The waveform (Figure 10) brings out all the five interfaces sending data through the entire interface at the same time. The figure depicts send buffers of all the interfaces getting a packet and request is raised by all the interfaces. On acknowledgment from the adjacent nodes, the packet is in the send registers for transfer. The nodes re interconnected to form the topology. The Figure 11 shows the RTL view of two interconnected nodes using Altera Quartus RTL viewer. The node is also tested with different FPGA family to verify the working. Table 1 tabulates the area occupied by the node (No of LE's) with its power consumption.

Table 1. Area Occupied In FPGA

FPGA	Area Consumed (No of LE's	Power Consumption (mW)
Cyclone II EP2C70F896C6	9.359	193
Arria II GZ EP2AGZ350FH29I3	7246	1149
Starix III EP3SL150F1152C3	7250	685

5.CONCLUSION

The paper presents in brief the RiCoBiT architecture. It mainly discusses the FPGA implementation of the same using verilog HDL. The papers discusses the internal blocks of a node which serves as a basic building block. Its implementation details are presented in the form of pseudo codes. The node is tested and verified for functionality using ModelSim and the same is presented as wave forms. The code was synthesised using Altera Quartus for different FPGA's, particularly for Cyclone II FPGA on DE2 70 board and real time parameters like area (No of LE's) and power consumed was recorded. The code was also subjected to real time testing using SignalTap. The nodes was interconnected to form the topology as depicted in the RTL diagram. It is observed that the node works correctly from the test conducted. As future work, we plan implementing different applications like matrix multiplication, security algorithms, cipher decoding etc using RiCoBiT architecture.

6. REFERENCES

[1] William J. Dally , Brian Towles, Route Packets Not Wires : on chip interconnection network, DAC 2001 June 2001.

[2] Tobias Bjerregaard And Shankar Mahadevan, A Survey of Research and Practices of Network on Chip, ACM Computing Survey March 2006..

[3] Ville Rantala, Teijo Lehtonen, Juha Plosila, Network On Chip Routing Algorithms, TUCS Technical Report No 779, August 2006.

[4] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johnny berg, Mikael Millberg,Dan Lindqvist, Network on a Chip: An architecture for billion transistor era, DAC 2001.

[5] Radu Marculescu, IEEE, Natalie Enright Jerger, Yatin Hoskote, Umit Y. Ogras and Li Shiuan Peh, Outstanding Research Problems in NoC Design: System, Micro architecture, and Circuit Perspectives, IEEE Transactions on computer aided design of integrated circuits and systems, vol. 28, no. 1, January 2009.

[6] J. Nurmi: Network on Chip: A New Paradigm for System_on_Chip Design. Proceedings 2005 International Symposium on System_on_Chip, 15 17 November 2005.

[7] L. Benini, G. De Micheli. Networks on chips: A new SoC paradigm. IEEE Computer. 35(1), 2002

[8] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M.Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, A Network on Chip Architecture and Design Methodology, Proceedings International Symposium VLSI (ISVLSI), pp. 117-124, 2002.

[9] Dan Marconett, A Survey of Architectural Design and Implementation Tradeoffs in Network on Chip Systems, University of California, Davis

[10] Nilanjan Banerjee, Praveen Vellanki, Karam S.Chatha, "A Power and Performance Model for Network-on-Chip Architectures," Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE) , p. 21250, 2004. Transactions on Computers ,vol. 54, no. 8, pp. 1025- 1040, August, 2005

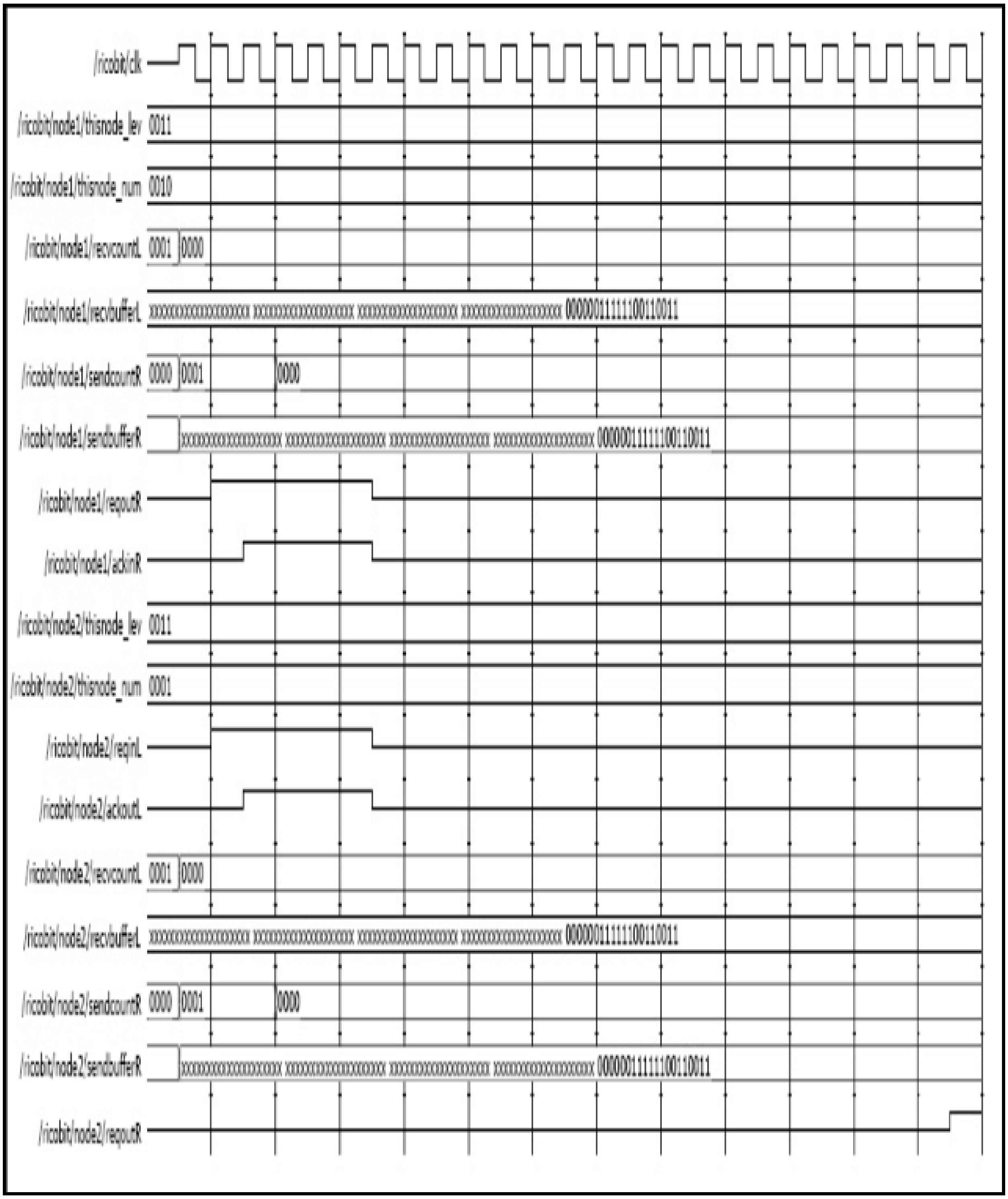


Figure 9. Waveform Depicting The Different Stages Of Communication Between Two Nodes

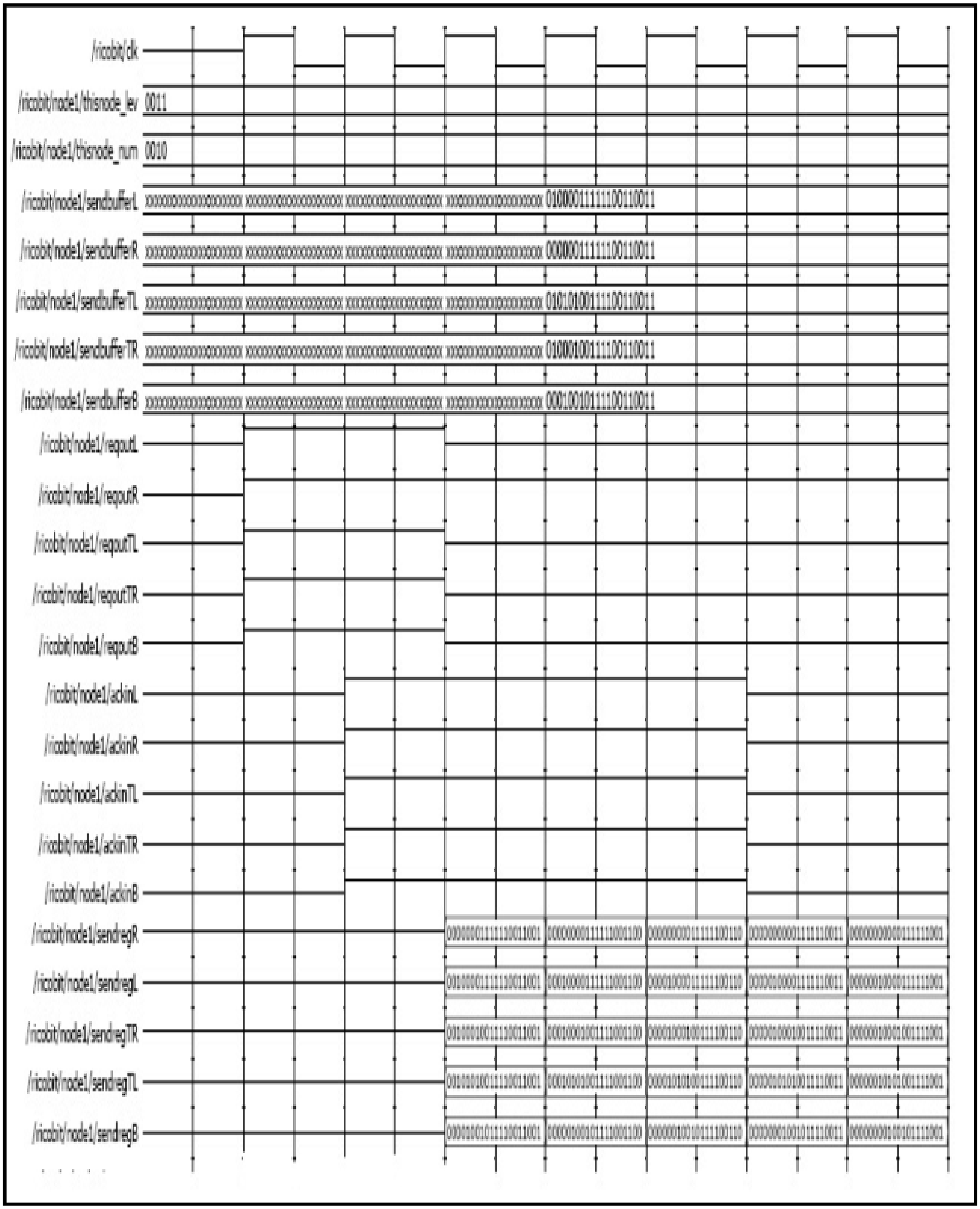


Figure 10. Waveform Depicting All Five Interfaces Of The Node Sending The Data

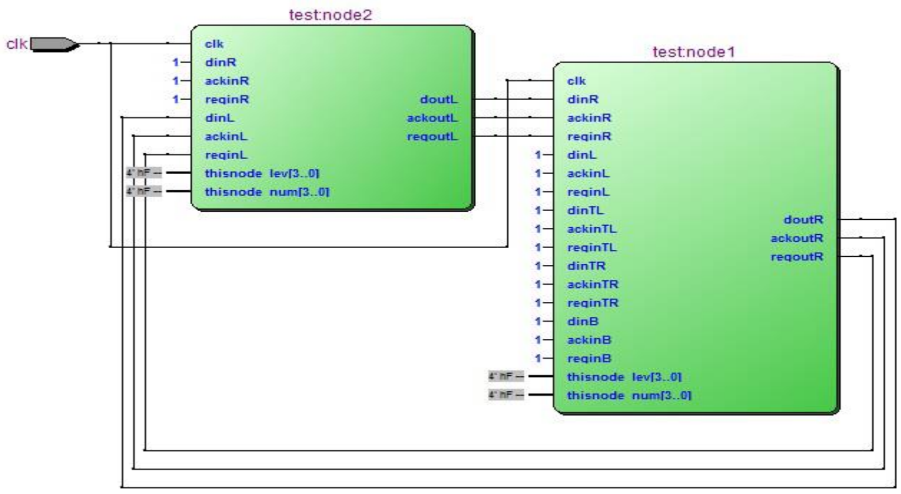


Figure 11. RTL Depicting Interconnection Of Two Nodes

Sanju V is working as Associate Professor with NMIT, Bangalore. He obtained his BE in Computer Science and Engineering from NMAMIT, Nitte, India in 2002, MTech in VLSI and Embedded Systems from Department of PG Studies, University Campus, VTU, Belgaum, India in 2005. And his PhD from VIT University, Vellore, India. His area of interest includes network on chip architectures, CAD for VLSI, and embedded systems.

Niranjan Chiplunkar obtained his BE in Electronics and Communication Engineering from National Institute of Engineering, Mysore in 1986, MTech in Computer Science and Engineering from Manipal Institute of Technology, Manipal in 1991 and PhD in Computer Science and Engineering from University of Mysore in 2001. His areas of interest include CAD for VLSI, computer networking and wireless body sensor networks. He has over 25 years of teaching and research experience. He is a member of IEEE, CSI, ISTE and Fellow of Institution of Engineers (India). He is currently working as the Principal, withNMAMIT, Nitte, India.

Suresh Mandia is a student at Warsaw University of Technology , Warsaw

Nancy Jain is a IT Engineer at CISCO, Bangalore, India

Nikita Jaiswal is a Anaylst at Mckinsey & Company, India

M. Khalid is a Senior Professor at the School of Computing Science and Engineering (CS&E),VIT University, Vellore, India. He was the Dean/Director for four plus years. He obtained his PhD (1981), MTech (1977) and BSc Engineering (1974) degrees from IIT Bombay, IIT Delhi and AMU Aligarh, respectively. He has a long-standing teaching, research and administrative experience in CS&E at IIT Bombay (1981–1998), Jordan (1996–1998 on lien), Qatar (1998–2003), and Bahrain (2003–2005). His research areas include high speed computer networks, design and analysis of protocols, performance evaluation of computing systems and fast packet switch architecture.