IJCSC Volume 4 • Number 2 September 2013 pp.145-152 ISSN-0973-7391

Design and Implementation of (N X N) Folded Torus Architecture for Network on Chip with E-Cube Routing

Phuntsog Toldan [1], Manish Kumar [2]
[1] National institute of electronics and Information Technology, Srinagar, J&K, India.
[2] Department of computer science, Galgotia college of engineering and Technolgy,Noida ,India.
toldhanleh@gmail.com, manish_sipu@gmail.com

Abstract
With the advancement in the field of computer technologies more and more computer components or intellectual property (IP) are integrated on a single chip thus making it more compact and complex. Systems on Chip' (SoC), the integration of multiple computer components on a single chip is a popular research field however communication between these on chip computer components has always been a center of attraction for the researchers. Communication between these on chips components were firstly done with Bus based system where a single bus was used to carry out the data between these components, crave for a better method and improvement over bus based technologies gave rises to Network on chip (NoC) design which Endeavour's to bring network communication methodologies to on chip communications, In hope of improving performance over current bus-based systems. A solution to flexibility and reconfigurability of interconnects are Network on Chip (NoC). These systems consist of a number of nodes (processors, memories, etc.) which are directly connected with each other through an on-chip network. The network is composed of a set of routers that handle communication between nodes. In this Paper, after studying thoroughly various Network on chip (NoC) based system which includes various architectures for Network on chip like Octagon, Torus etc, various switching techniques and routing algorithms and after a detailed study of the various architectures I propose a new architecture called Folded Torus architecture as a network topology which uses Ecube routing algorithm to route data from one component to another. Folded Torus architecture was introduced as an improvement over torus topology which is discussed in detail in the report. The thesis also studies the project with different network dimensions and analysis the results to see which dimension works with the project best. We uses an application called GTKWave to study the signals generated by the project and another application called Gossip application to study whether the project is working properly or not.
Keywords: System on chip, Network on chip, Folded Torus network, Ecube routing

1. Introduction

In the area of Embedded System, There is a desire to create 'Systems on Chip' (SoC). The Integration of multiple computer component or entire electronic systems (Microcontroller, Memory Blocks, Timers etc) on a Single chip. However, a new and increasingly popular Research Field which addresses some of the concerns of effective communication between on chip components, Network on chip (NoC) design, Endeavour's to bring network communication methodologies to on chip communications, In hope of improving performance over current bus-based systems.

The basic concept is to communicate across the chip in the same way that messages are transmitted over the Internet today. That is, put a packet switching Network on the chip and send messages back and forth between blocks.

There are various techniques proposed by many researchers but craving for a better one has always been a habit of scientific research. The technique discussed here uses folded torus architecture as the network topology with Ecube routing. The folded torus topology uses mips processor as its nodes, each mips processor has a 2 dimensional router (XY router) connected to it which does the work of forwarding the data to other routers in accordance with Ecube routing algorithm. This base of this paper is the implementation of (2 x 2) torus topology with Ecube routing algorithm which make use of the torus topology.

The router and the resources (MIPS processor) as well as the torus topology are designed in systemc language which is a library of c++ classes.

The main objective of this thesis is to generalize the miniNoC project [13] i.e. generalizing the 2x2 miniNoC network into NxN network and thus enabling the user to work with as number of processor as he wants. Also using a folded Torus topology instead of the Torus topology which was used in the miniNoC project which according to the researches is more efficient.

2. Present Work

The various steps used in the Thesis are discussed below.

2.1 Introduction

The main objective of this thesis is to generalized the miniNoC(network on chip) [13], where a 2x2 network is used on a torus network with ecube routing. In this thesis the torus network is extended to folded torus network and the network dimensions are generalized to NxN. The folded torus topology uses mini-mips processor as its nodes, each mips processor has a 2 dimensional router (XY router) connected to it which does the work of forwarding the data to other routers in accordance with Ecube routing algorithm. The NxN network is introduced to bring flexibility in the miniNoC network so that the user can work with as number of processors as he wants. The main part is the generalization of the network and using the folded torus network instead of the torus topology.

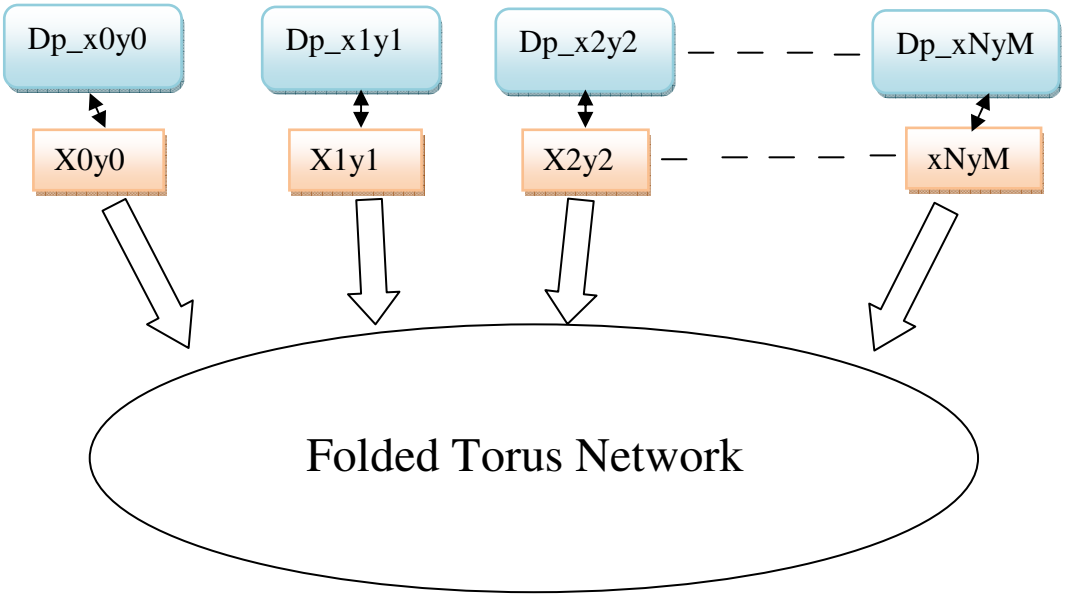

Figure 1: NxN Folded Torus architecture for network on chip (NoC)

The router and the resources (MIPS processor) as well as the torus topology are designed in systemc language which is a library of c++ classes.

2.2 Folded Torus Network

The Folded Torus network is similar to the Mesh architecture except that the wires are wrapped around from the top component to the bottom and the rightmost component to the leftmost, Thereby doubling the bandwidth of a mesh network, The folded torus network is actually a torus Network with a difference that in torus network The rightmost node loops back and connects the leftmost node which produces an overhead in time taken to pass the data between the rightmost to leftmost or vice versa than to pass data between other two successive nodes, Whereas in Folded Torus network the alignment is such that time taken to pass data between every node is same shown in the figure 2. Although the torus architecture reduces the network diameter, the long wrap-around connections may result in excessive delay. However, this problem can be avoided by folding the torus, as illustrated in Figure 2 to make it a folded torus.

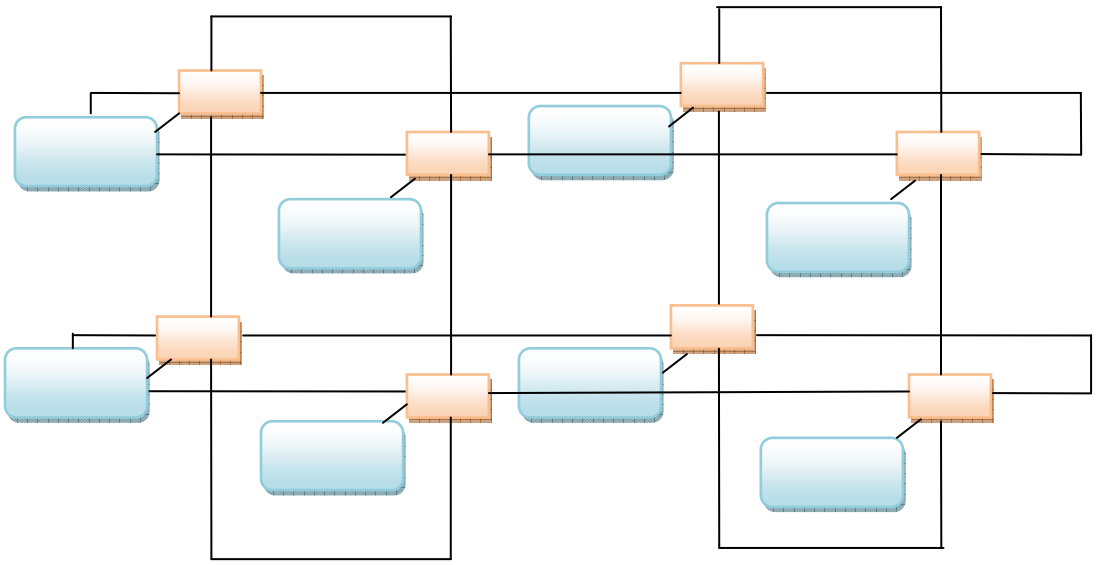

Figure 2: A Folded Torus Network

2.3 Ecube Routing Algorithm

Ecube routing is a data routing mechanism in which each packet is routed 1st along a particular dimension until it reaches a router with address equals to that packets destination address and then it moves along the next dimension and so on. for example if we take a network with 2 dimension, then the packet is 1st routed along the X dimension until it reaches a router with the X address equal to the packets destination X address. Then it starts moving in Y dimension until it reaches the router with the Y address equal to packets destinations Y address and hence the destination router. In the given figure the packet is currently at the node [3,1] and it is destined to the node [2,2], now according to the Ecube routing algorithm the packet will 1st travel in the 1st dimension until it finds the x=2 and then it travels in the 2nd dimension until it gets y=2.

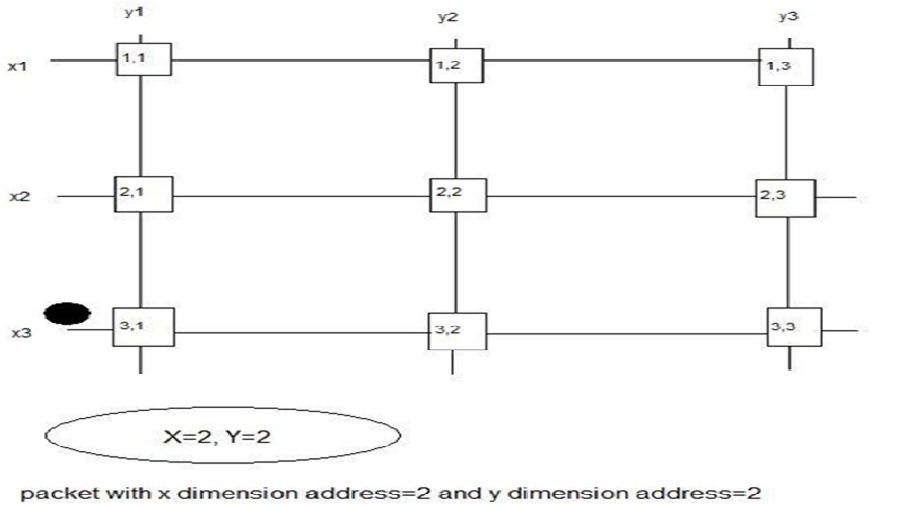

Figure:3 Ecube routing algorithm

2.4 Need for generalization the network.

mMIPS Network on chip of miniNoC project [13] from where the main refence have been taken deals with Network on chip designed on a 2x2 network i.e. a network with 4 nodes. It has four MIPS processors attached to four Routers which in turn are attached to one another in order to carry on the process of transmission of data.
In this paper i have generalized the miniNoC network , i.e. now the network will have N numbers of processor and routers, where N is specified by the user and it can range to any number of nodes.
The 2x2 network is a inflexible network where the user can work with only 4 nodes if the user wants to increase the number of nodes or if he wants to work with more number of nodes then he has to manually manipulate the code thus is quite inflexible.
Thus in this thesis i have converted the miniNoC into a flexible network by converting the network from 2x2 into a NxN, where N can range to any number. Thus now the user have the flexibility to work with as many number of nodes as he wants.

2.5 Making NxN Network
Generalizing the miniNoC includes some manipulation with the codes, the miniNoC code includes various important modules like the 2X2 network, the router, the processor (mMIPS), cache memory and various other small modules, to genalize the network i.e. to make it into a genalized NxN network we mainly manipulate the network module which in the mininoc project is named as network2x2.cpp and network2x2.h.
The network used in the mNoC is composed for four nodes with one mMIPS processor on each connected using a torus network with E-cube routing. In order to achieve the main objective for this project, it was necessary to generalize the network into N number of nodes allowing the processors to handle more data in this way. For this reason, the network has been generalized from 4 nodes to N, which means a transition from a NETWORK2x2 to either NETWORK_NxN. The rest of the modules are kept the same, the cache memory module, RAM module, ROM module, Register modules are all kept unchanged and used here the way they were used in mininoc platform.
The most important thing to do this was to study the mininoc project carefully and thoroughly, the second thing was to recognize and differentiate the main modules on which we have to work (The Processor mMIPS and The Router). After doing so we separate each of the functions and make them work separately to see whether they are fine or not i.e. take a single instance of each of the module mMIPS and Router, connect them and finally test them. Since we have a single instance of each of the module now we will try to generalize each module i.e. making N number of modules of each type. To do that we use arrays with size of N where N can be of any range.

2.6 Implementation
Generalizing the network needs manipulating the miniNoC source code and converting the 2x2 network into a generalized NxN network. In 2x2 network four instances of each Router and the processor (mMIPS) are created and are connected with each other in order to carry on the required operations.
Here in this thesis i have taken arrays to denote the routers as well as the processor (mMIPS). i.e. creating N Routers by creating a Router array of size N and creating N processors by creating a processor (mMIPS) array of size N.
To be able to achieve the diagram shown in the previous figure, the first change was made in the Systemc description of the network. N number of routers are created by using arrays (xy[n][n]) to obtain a networNxN. Similarly N number of mMIPS processor are created by using arrays (dp_xy[n][n]).

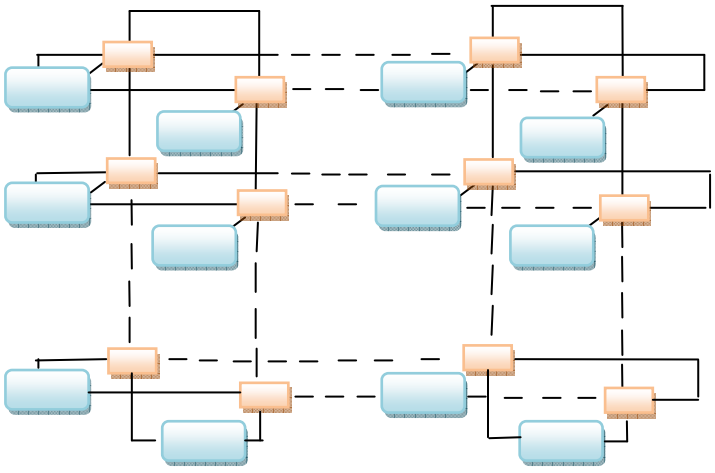

Figure 4: A NxN Folded Torus Network

xy[n][n] is a N-dimensional Router where n is defind by the user and can be extended to any number and dp_xy[n][n] is a N-dimensional mMIPS processor.
The figure below shows a piece of code where N number of 2-dimensional router is created namely xy[xdim][ydim]; where xdim and ydim are abbreviations for x-dimension and y dimension and can range to any value as specified by the user. ECUBE_ROUTER is a module in which the 2 dimensional router is created, and xy[xdim][ydim] is therefore object of ECUBE_ROUTER module. Thus by specifying any values to xdim and ydim we can create any number of 2-dimensional routers for example xy[0][0], xy[0][1], xy[1][0] and so on. The processor mMIPS are created in a similar fashion in which N number of mMIPS is created using arrays and are connected with each router to carry on the specified tasks. NETMIPS is a module in which all the signals, input output ports and variables of the MIPS processor are declared and we create N numbers of mMIPS by creating N number of objects of NETMIPS type just like the way we did for Routers.

```
ECUBE_ROUTER *xy[xdim][ydim];
NETmMIPS *dp_xy[xdim][ydim];

for( i=0; i<=xdim; i++ )
    for( j=0; j<=ydim; j++ )
        {
         xy[i][j] = new ECUBE_ROUTER("xy[i][j]");

         dp_xy[i][j] = new NETmMIPS("dp_xy[i][j]");

          xy[i][j]-> clk(clk);
          xy[i][j]-> rst(rst);
          xy[i][j]-> din(xydin[i][j]);
          xy[i][j]-> dout(xydout[i][j]);
          xy[i][j]-> dreq_in(xyreq_net[i][j]);
          xy[i][j]-> dack_out(xyack_net[i][j]);
          xy[i][j]-> dreq_out(xyreq_dp[i][j]);
          xy[i][j]-> dack_in(xyack_dp[i][j]);
        }
```

Figure 5: creating instances of the Router

3. Results

Compiling and running the whole project creates various signals which can be viewed in GTKwave application. Unlike the mininoc project which created signals for only four nodes this folded torus project creates signals for as many number of nodes as the user wants. The user just has to specify the number of nodes he would be using before compilation of the project.

Below is a figure which shows a piece of simulation code. The code executes all the nodes and after executing them displays a message finishing messages with the respective node numbers (for example Finished 00, Finished 01, Finished 02 etc).

Figure 4.7 below shows a comparative study between different dimensions taken with respect to the clock cycles taken to execute them. After doing a detailed study we came to kno that the clock cycle taken to execute the network

```
for(int i=0;i<=xdim;i++)
    for(int j=0;j<=ydim;j++)
     {
      int k=i;
      int l=j;
       if( !e[i][j] &&
       dp_xy_pc[i][j].read().to_uint() == 0x28 )
       {
       cout << "FINISHED" << k<< l<< "@"  <<
       sc_time_stamp() << endl;
       e[i][j] = true;
        break;
       }
      }
```

Figure 6: piece of simulation code

with a particlular dimension decreses as we increse the network size but when the network dimension reaches to 12x12 it becomes stagnent even after we increase the dimension this shows that the project works well till a particular point and then it performance becomes constant for all other higher dimensions.

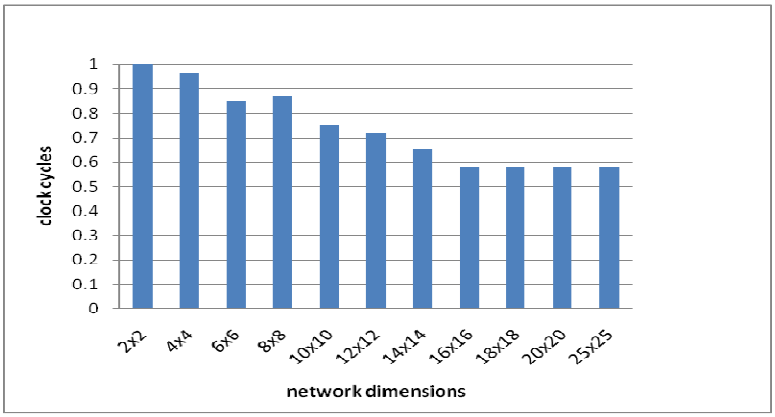

Figure 7: Comparison of different network dimension

4. Applications

Gossip is a Test application which is being used to test the functionality of the original mMIPS. It is a very simple program that sends a short message across the network. This application can be used in the same way it was used for the original mNoC project.

4.1 Test application Gossip

Gossip is an application to test if the whole system has been set up correctly, simply sending a short message across the network.

Node 00 send the text "*I know something!*" to node 10 and then listens for a return message from any node. When it has received that message, it quits. All other nodes wait for an incoming message; forward it to the next node and then exit.

After executing this extended Gossip, the content of the RAM memory should be examined in order to check whether network functionality is correct or not. The Debug Info area of this memory (see *3.1.2*) contains information about the application execution. In the case of a NETWORK2x2, the result was the following:

```
NODE 00:
=========
Node 0 up and running!
I sent:    "I know something!"
I received:"I know something!"
Let's call it a day!

NODE 10:
=========
Node 1 up and running!
I received: "I know something!"
Successfully forwarded the message.
Let's call it a day!

NODE 01:
=========
Node 2 up and running!
I received: "I know something!"
Successfully forwarded the message.
Let's call it a day!

NODE 11:
=========
Node 3 up and running!
I received: "I know something!"
Successfully forwarded the message.
Let's call it a day!
```

Figure 8: Gossip debug info for 4 nodes

5. Conclusion

As it has been said in this report, many changes had to be done in the original mNoC in order to achieve the main objective proposed for this project. The original mNoC is designed on a simple torus network which has only four nodes, i.e. it is a simple 2x2 torus network which is the base of this thesis work. Torus topology was proposed to reduce the latency of mesh and keep its simplicity. The only difference between torus and mesh topology is that the switches on the edges are connected to the switches on the opposite edges through wrap-around channels although the torus architecture reduces the network diameter, the long wrap-around connections may result in excessive delay. However, this problem can be avoided by Folding the torus. Thus the use of folded torus topology incurs less or no delay and better utilization of the network.

This project generalizes the original mNoC project from a 2x2 network to an NxN network so as to increase the functionality of the project. Generalizing the network makes the project flexible and the user can work with as number of nodes as he wants just by specifying in the beginning.

This project also introduces a new network topology called folded torus network unlike the one used in the original mNoC which is a simple torus network. The folded torus architecture was chosen as it provides a convenient and better topology.

REFRENCES

1. Karim.F, Nguyen.A, Dey.S "An interconnect architecture for networking systems on chips" Proceedings of IEEE Sep/Oct 2002 Volume:22 Issue:5
2. Hemani.A, Jantsch.A, Kumar.S, Postula.A "Network on chip: An architecture for billion transistor era." In Proceedings of the IEEE NorChip Conference, (Nov 2000)
3. Obias.B and Shankar.M "A survey of research and practices of Network on chip" ACM Computing surveys, Vol.38, March 2006.
4. Luca.B, Giovanni de.M, "Network on chips: A new Soc paradigm," Computer.v.35 n.l, p.70-78, Jan 2002
5. Muhammad.A, Micheal.W, "A dynamic Routing mechanism for network on chips," Proceedings, 23rd NorChip conference, Nov. 2005, Finland
6. W.J. Dally and B. Towles, "Route Packets, Not Wires: On-ChipInterconnection Networks," Proc. Design Automation Conf. (DAC), pp. 683-689, 2001.
7. J. Duato, S. Yalamanchili, and L. Ni, "Interconnection Networks—An Engineering Approach," Morgan Kaufmann, 2002.
8. S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks," IEEE Transactions on Computers, vol. C-38, no. 4,pp. 555– 566, April 1989.
9. F. Karim, A. Nguyen, and S. Dey, "An Interconnect Architecture for Networking Systems on Chip," IEEE Micro, vol. 22, no. 5, pp 36–45, September/October 2002.
10. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "Spin: A scalable, Packet Switched, on Chip Micro-Network," In DATE 03 Embedded Software Forum, pages 70–73, 2003.
11. W.J. Dally and C.L. Seitz, "The Torus Routing Chip," Technical Report 5208:TR: 86, Computer Science Dept., California Inst. of Technology, pp. 1-19, 1986.
12. Dan Marconett "A survey of architectural design and implementation Tradeoffs in network on chip systems"
13. A. Slusarczyk, S. Stuijk, M. Visser. "mMIPS Network-on-chip website". http://www.es.ele.tue.nl/mininoc/doc/module_network2x2.htm
14. Kun Wang, Huaxi Gu, Changshan Wang "Study on Hybrid Switching Mechanism in Network on Chip" School of computer science, Xidian University, Xi'an and State Key Lab of ISN, Xidian University, Xi'an china
15. A. Jantsch, H. Tenhunen. "Networks on Chip". *Kluwer Academic Publishers*. February 2003
16. Synopsys, Inc. "SystemC v2.0 User´s Guide". 2002