NoCGIN: A Gamma Interconnection Network as NoC Interconnect

Meenal A. Borkar
Research Scholar
School of Engineering and Technology, Ansal University, Gurgaon, India

Nitin Nitin
Associate Professor
Department of CSE and IT, Jaypee Institute of Information Technology, A-10, Sector-62, Noida-201307, India

Atul Kumar
Professor
School of Engineering and Technology, Ansal University, Gurgaon, India

ABSTRACT
As billions of transistors can easily getting manufactured on small chips, multiple processing elements are also getting fabricated on these chips. This type of chip manufacturing caught attention of researchers from the domains like Parallel and Distributed Computing, Computer Aided Chip Manufacturing, Computer Design etc. Many researchers tried to utilize the boosted capacity of multiprocessor chips to implement time consuming, bulky, parallel algorithms. A strong communication network, which is reliable, robust and reusable is very much needed to achieve expected performance. This paper proposes a new Gamma Interconnection Network variant, namely NoCGIN, which act as interconnection network for Networks-on-Chip. The paper further gives information about the topology of NoCGIN and a simple routing algorithm for routing packets.

Keywords
Networks-on-chip, Gamma Interconnection Network, Systems-on-chip, Parallel Computing

1. INTRODUCTION
In this era of high speed processing, multiple processors are fabricated on single chip. To achieve speed, the computations are distributed among these processors. The processors are interconnected to each other using either direct links, crossbars or multistage interconnection networks. The multistage interconnection networks prove economical as well as beneficial and provide good connectivity with increased reliability. Majority of the researchers worked on multistage interconnection networks to provide higher reliability, so that the high end processing systems become more robust for parallel and distributed processing. One of the challenges faced by theses researcher was, designing MINs which are not fault robust. By fault robust, we mean, if the fault exists then communication between some processors get abandoned. The issue was handled in past by designing the networks, which are capable of providing multiple paths or redundant paths[1-7].

1.1 Interconnection Networks
Interconnection Networks (IN) [1-7] have a very rich history. Figure 1 shows the typical structure of IN. The development of IN can be attributed to its use in three major areas, namely

- Telephone Networks
- Inter-processor communication networks
- Processor-memory interconnection networks

Telephone networks were using INs since their conception. In earlier days, electro-mechanical crossbars and step-by-step switches were used. The major research developments in telephone networks were non-blocking networks, multistage Clos networks and Benes networks. By 1980, long-distance calls were made using digital and electronic switches, whereas local calls were made using electro-mechanical switches [4][6].

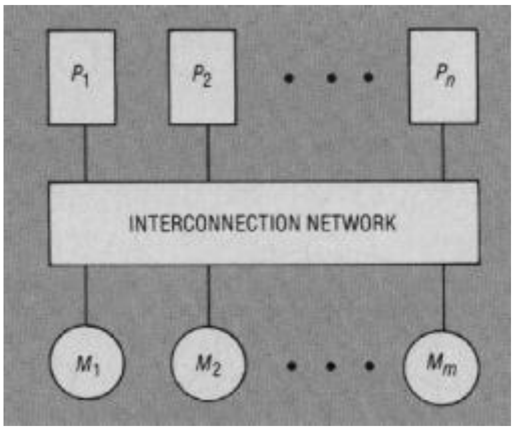

Fig 1: Interconnection Network

Inter-processor interconnection network came into the picture when the processors needed to be connected in a 2D array style. Initial machines like Solomon, Illiac and MPP were based on simple INs like 2D mesh or torus. These INs were preferred because the initial machines required physical regularity in interconnections. Binary n-cube and hypercube became popular, due to their low diameter, in late 1970s. The low dimension networks were found performing quite well under realistic packaging constraint; so many manufacturers again started using 2D and 3D topologies.

Processor-memory interconnections emerged in late 1960s and were used to allow parallel processors to access the memory without burdening the other processors. Crossbar switches are widely used for these types of system interconnects.

These three development threads together are used to design the interconnection network in modern systems. Designing an interconnection network for parallel processors with multiple memory banks is always a critical task. To achieve faster access to memory units without introducing much waiting, multistage interconnection networks (MIN) [4-7] were invented. After 1980s, a lot of research was carried out to satisfy the needs of the demanding communication problems of multi-computers. This research was driven by developments in the technology to construct single–chip Very

Large Scale Integration (VLSI) routers. Then, a series of new ideas filled the research jargon of digital communication systems.

1.2 Multistage Interconnection Networks

Due to increased use of multiprocessor systems the reliability, availability and performance characteristics of the networks that interconnect processors to processors, processors to memories and memories to memories captured the attention of researchers. A Multistage Interconnection Network (MIN), in particular, is an IN consisting of a cascade of switching stages, each containing switching elements (SE). Figure 2 shows a typical MIN, connecting processors to memory units. MINs are widely used for broadband switching technology and for multiprocessor systems. Besides this, MINs offer an effective method of implementing switches used in data communication networks. With performance requirement of the switches exceeding several terabits/sec and teraflops/sec, it becomes very necessary to make them dynamic and fault–tolerant[8-13].

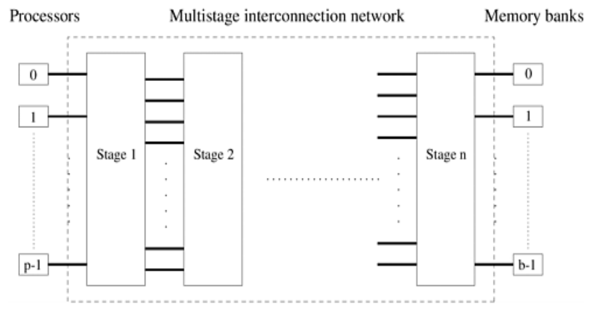

Fig 2: Multistage Interconnection Network

1.3 Gamma Interconnection Network

The Gamma Network [14-15] is a multistage interconnection network, which uses the redundant paths method for design. It connects N = 2n inputs to N outputs. It consists of (log2N) + 1 stages with N switches per stage. These switches are connected with each other using 3 X 3 crossbar switch. The input stage uses 1 X 3 crossbar, output switch uses 3 X 1 crossbar and all the intermediate switches use 3 X 3 crossbar. A sample Gamma Network is shown in Figure 3. The stages are linked together using “power of two” and identify connections such that redundant paths exist. The path between any source to destination is represented using any one of the redundant forms of the difference between source and destination. These redundant forms are generated using Binary Redundant Number System[28].

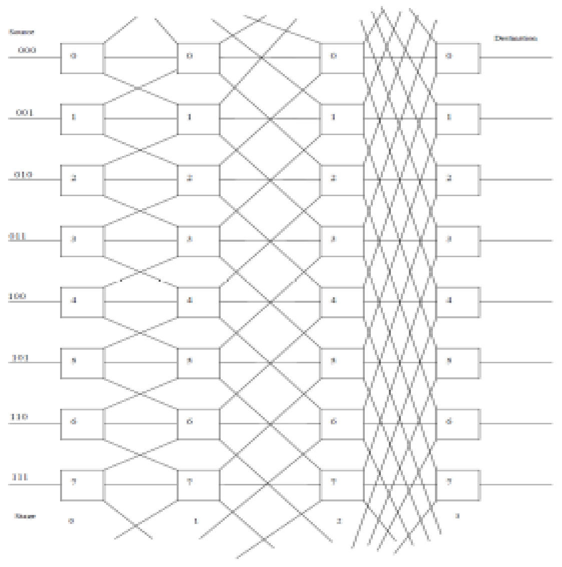

Fig 3: Gamma Network

The Gamma Network uses binary redundant form [23][28] of difference between source and destination. This form is better known as tag or routing tag. A bit in routing tag can take three values: 1, 0 and -1. The routing tag T = (tn-1, tn-2, t0), where the first bit is MSB and the nth bit is LSB. There are three possible interconnections possible at a stage i. The data from switch j takes straight path to deliver data to switch j in stage i+1, take upward path to reach switch (j – 2i) mod N and take downward path to reach switch (j + 2i)mod N. The Gamma Network can realize perfect shuffle, cyclic shifts and permutation shifts. Researchers tried various ways to provide fault tolerance to GIN. There are 20 plus network variants available in literature[15-35]. Table 1 lists the network variants along with the routing methods compatible with each of them. Interested readers can find the detailed information about Gamma Interconnection Network Family in [35].

Table 1. Gamma Network variants with their routing methods

Sr. No.	Name of Network	Routing Method Used
1.	GIN	Distance Tag Routing
2.	Kappa Network	Destination Tag Routing
3.	Extra Stage GIN	Distance Tag Routing
4.	B-Network	Destination Tag Routing
5.	Balanced GIN	Distance Tag Routing
6.	Mono GIN	Distance Tag Routing
7.	Reliable GIN	Distance tag Routing
8.	Cyclic GIN	Distance Tag Routing , Destination Tag Routing
9.	Partially Chained GIN	Distance Tag Routing
10.	Fully Chained GIN	Distance Tag Routing
11.	3D GIN	Distance Tag
12.	3D-CGIN	Distance Tag Routing, Destination Tag Routing
13.	Incomplete GIN	Twin Tag Routing based on Distance Tag Routing

14.	Incomplete CGIN	Twin Tag Routing based on Distance Tag Routing
15.	4DGIN-1	Distance Tag Routing
16.	4DGIN-2	Distance Tag Routing
17.	NBGIN	Destination Tag Routing
18.	MCDRGN(Additional link at initial stage)	Distance Tag Routing
19.	MCDRGN (Additional link at initial and intermediate stages)	Distance Tag Routing
20.	DRGIN	Distance Tag Routing, Destination Tag Routing
21	GIN with Alternate Source	Distance Tag Routing, Destination Tag Routing

1.4 Networks-on-Chip

With advances in VLSI technology, fabricating thousands of circuits on small area is a common practice. Since last decade, the chip manufacturers used the advanced technology to map multiple cores on a single chip. Today, 4 to 8 core processors is normal processing configuration. To provide higher processing capability, the chips with 64[36], 80[37] and 100[38][40] cores are also available. Researchers have also manufactured, chips with 1000[39][40][42] cores as a research prototype for High Performance Processing. When multiple cores are available on single chip, depending on the application scheduled for processing, communicate with each other. During communication, these cores exchange data and information. There is a need for providing some kind of network for these cores. The interconnection network used for this purpose is known as Network on Chip (NoC). The traditional bus system was used initially, as it is the cheapest topology to implement, for this purpose, but it was found that- (1) the speed of communication was slow, (2) suffers from effects of crosstalk and electromagnetic interference, (3) if tried to scale beyond a particular number, hampers the communication[41][45-47][50-51][58-60][66-68].

To overcome these problems, researchers started checking and using the applicability of other Interconnection Network topologies. The Mesh and Torus topologies are popularly used as NoC interconnects. The detailed information can be referred from [48-49][52-55][57][62-65].

While studying the available literature, we found multiple MINs and INs used as NoC interconnects. When started looking for similar implementation / use of Gamma Network, we found nothing. The Gamma Network when proposed found suitable for implementation of Fast Fourier Transform(FFT) algorithms, which means it has inherent design capability to work as NoC interconnect. During literature review, it was also observed that, with such a variety of network variants, with improved path generation capabilities, GIN can prove better interconnect for NoCs. This motivated us to start our work in that direction.

This paper is organized as follows: Section 1 – presented the basic information about INs, MINs, GIN and NoC. The section also presented the motivation for this work. Section 2 – presents the proposed work, specifically a new variant of GIN namely NoCGIN. We present the topology of NoCGIN with simple routing algorithm. Section 3 – presents the experimental setup, assumptions and conclusion. Section 4 – presents the future work scope, followed by the references.

We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace the content with your own material.

2. PROPOSED WORK

In this section, we present the topology of NoCGIN and the routing algorithm for it. The section also provides suitable examples to demonstrate the routing.

2.1 Topology of NoCGIN

NoCGIN is a Gamma Interconnection Network of size $N = 2^n$. Here N is the number of inputs and number of outputs it is connecting with. The NoCGIN will have $(\log_2 N)+1$ number of stages, numbered from 0 to $(\log_2 N)+1$. Each stage will have N number of switching elements(SE). The SEs in stages will have one input from core it is connected with and 3 bidirectional links. The bidirectional links connect the SE to 2 SEs in next stage and the immediate next SE in same stage. For the SEs in last stage, the bidirectional links connect to previous stage and the immediate next SE in same stage. Figure 4 shows the typical topology of NoCGIN of size $N = 2^2$. The NoCGIN as a variant of GIN is compatible with Distance Tag Routing.

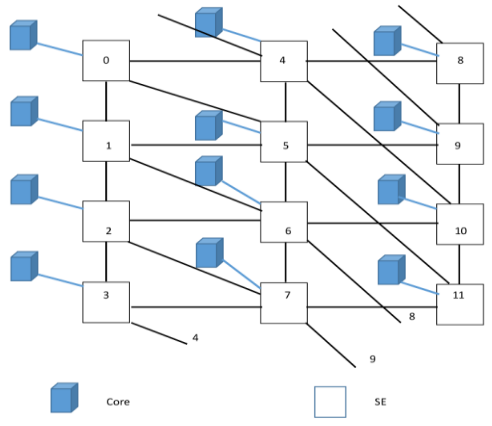

Fig 4: NoCGIN Topology size N = 4

2.2 Working and Routing Algorithm of NoCGIN

As we can see in Figure 4, the SEs are connected with cores and other SEs with bidirectional links. The cores generate the messages for communication. These messages are then divided into smaller entities known as packets. Each packet carries the address of source core number and destination core number. The data inside the message is known as payload. Therefore a packet's format is similar to one shown in Figure 5.

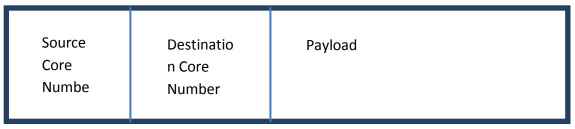

Fig 5: Format of Packet

The general strategy for packet routing is as follows:

1. If the source and destination are from same stage, use the vertical links to route packet.

2. If the source and destination are located in initial / end stages, use Destination Tag Routing, where the tag is made up of 2 bits,
3. If the source and destination are located in immediate next or previous stages, use Destination Tag Routing, where the tag is made up of 1 bit.

The detailed implementation of this strategy is listed as an algorithm in Table 2 given below.

Table 2. Routing Algorithm for NoCGIN size N = 4

```
Algorithm: Routing in NoCGIN

Input: Packet with source, destination and payload

Output: Step by step packet routing

Method:

Let N = 4

1. Take the packet for routing,

2. if (source and destination are in same stage) then
       if (source < destination) then
           Use the vertical link in down direction until the destination is reached.
       else
           Use the vertical link in up direction until the destination is reached.
       [End of If]
   [End of Step 2 If]

3. if (source is in stage 0 and destination is in stage 2) then
     i. calculate the difference as
        diff = (destination)mod N – (source)mod N
     ii. Generate Two bit tag for diff
     iii. if (the tag generation is not possible) then
          a. if((source)mod N > (destination) mod N) then
                Use the vertical link in upward direction
             Else
                Use the vertical link in downward direction
             [End of if]
          b. Let newsrc = The SE reached using this will act as new source
          c. Goto step 2 , where source = newsrc
        [End of Step iii if]
     iv. Use this tag in reverse direction to route packet.
   [End of step 3 If]

4. if (source is in stage 2 and destination is in stage 1) then
     i. calculate the difference as
        diff = (destination)mod N – (source)mod N
     ii. Generate Two bit tag for diff
     iii. if (the tag generation is not possible) then
          a. if((source)mod N > (destination) mod N) then
                Use the vertical link in upward direction
             else
                Use the vertical link in downward direction
             [End of if]
          b. Let newsrc = The SE reached using this will act as new source
          c. Goto step 2 , where source = newsrc
        [End of Step iii if]
     iv. Replace every 1 with -1, which indicates the cross link is to be followed in back direction
     v. Use this tag in reverse direction to route packet.
   [End of Step 4 If]

5. if (source is in stage 0 and destination is in stage 1) then
     i. calculate the difference as
        diff = (destination)mod N – (source)mod N
     ii. Generate One bit tag for diff
     iii. if (the tag generation is not possible) then
          a. if((source)mod N > (destination) mod N) then
                Use the vertical link in upward direction
             else
                Use the vertical link in downward direction
             [End of if]
          b. Let newsrc = The SE reached using this will act as new source
          c. Goto step 2 , where source = newsrc
```

```
    **[End of Step iii if]**

    iv. Use this tag to route packet.

[End of Step 5 If]

6. if (source is in stage 1 and destination is in stage 0) then

    i. calculate the difference as

        diff = (destination)mod N – (source)mod N

    ii. Generate One bit tag for diff

    iii. if (the tag generation is not possible) then

        a. if((source)mod N > (destination) mod N) then
            Use the vertical link in upward direction

            else

            Use the vertical link in downward direction

        [End of if]

        b. Let newsrc = The SE reached using this will act as new source
        c. Goto step 2 , where source = newsrc
    [End of Step iii if]

    iv. Replace every 1 with -1, which indicates the cross link is to be followed in back direction

    v. Use this tag to route packet.

[End of Step 6 If]

7. if (source is in stage 1 and destination is in stage 2) then

    i. calculate the difference as

        diff = (destination)mod N – (source)mod N

    ii. Generate One bit tag for diff

    iii. if (the tag generation is not possible) then

        a. if((source)mod N > (destination) mod N) then
            Use the vertical link in upward direction

            else

            Use the vertical link in downward direction

        [End of if]

        b. Let newsrc = The SE reached using this will act as new source
        c. Goto step 2 , where source = newsrc
    [End of Step iii if]

    iv. Use this tag to route packet.

[End of Step 7 If]

8. if (source is in stage 2 and destination is in stage 1) then

    i. calculate the difference as

        diff = (destination)mod N – (source)mod N

    ii. Generate One bit tag for diff

    iii. if (the tag generation is not possible) then

        a. if((source)mod N > (destination) mod N) then
            Use the vertical link in upward direction

            else

            Use the vertical link in downward direction

        [End of if]

        b. Let newsrc = The SE reached using this will act as new source
        c. Goto step 2 , where source = newsrc
    [End of Step iii if]

    iv. Replace every 1 with -1, which indicates the cross link is to be followed in back direction

    v. Use this tag to route packet.

[End of Step 8 If]
```

This algorithm takes care of all the possible cases,where the tag can not be generated using Distance Tag Routing. The interested users can refer to [Parker and Raghavendra paper] to get tag generation formula for distance tag routing. Let us take few examples to demonstrate the working of this algorithm.

Example 1: Let us assume Source = 0 and Destination = 3. Now as we can see, the source and destination are in same stage. So step 2 will come in picture. Next we will check whether source < destination, which is the case here. We will start following the vertical straight link in down direction to reach SE 1. Again the destination is not reached, and both source and destination are from same stage. New source 1 < destination 3, so again the vertical straight link is followed in down direction. The process repeats until destination 3 is reached. Figure 6 shows this routing.

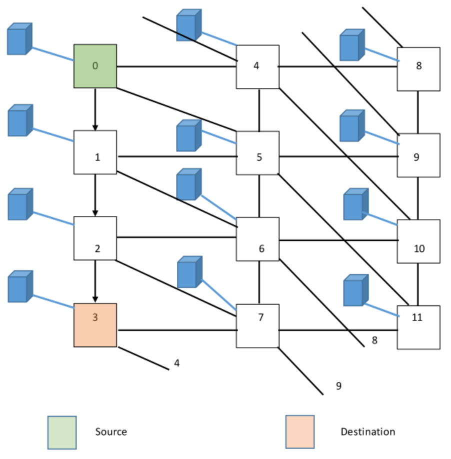

Fig 6: Routing packet from 0 to 3 using Step 2 of the algorithm

Example 2: Now suppose the source is 7 and destination is 5. Again both, the source and destination are in same stage so Step 2 is followed. The vertical link in upward direction is traversed until the destination 5 is reached. Figure 7 shows this routing.

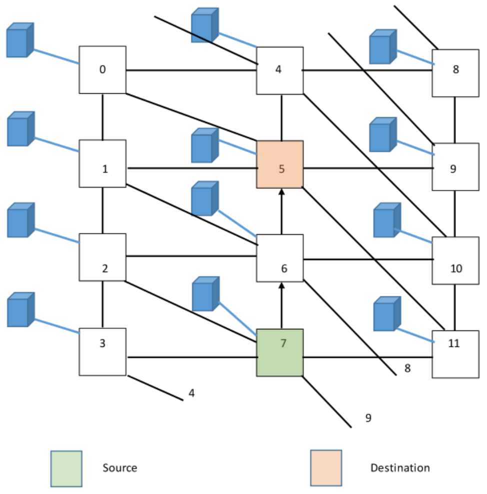

Fig 7: Routing packet from 7 to 5 using Step 2 of the algorithm

Example 3: The source = 2 and destination = 10. The source is in stage 0 and destination is in stage 2, therefore Step 3 is used. Here the difference is calculated as diff = (10)mod 4 – (2)mod 4, which comes out to be 0. The tag generated in this case is 00. We will use this tag in reverse order to reach destination. Figure 8 shows the routing.

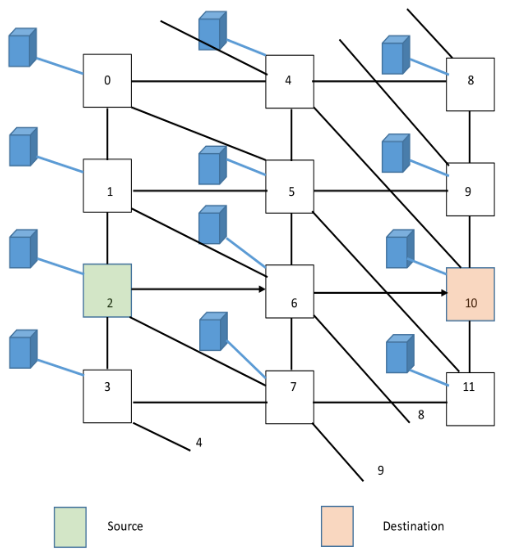

Fig 8: Routing packet from 2 to 10 using Step 3

Example 4: The source = 9 and destination = 2. The source is in stage 2 and destination is in stage 0, therefore Step 4 is used. Here the difference is calculated as diff = (2)mod 4 – (9)mod 4, which comes out to be -1. The tag generated in this case is 1-1. Now we need to replace each 1 by -1. So the modified tag becomes -1-1. We will use this tag in reverse order to reach the destination. Figure 9 shows the routing.

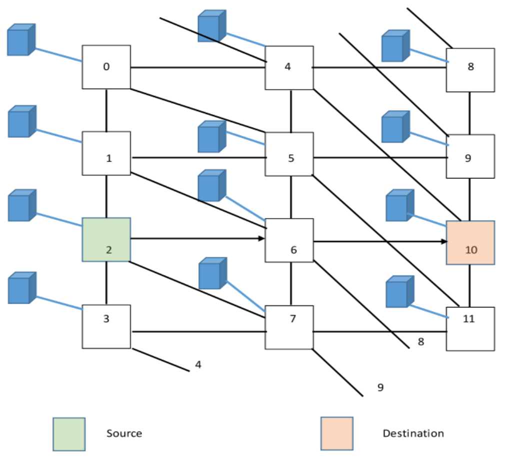

Fig 9: Routing packet from 9 to 2 using Step 4

Example 5: The source = 1 and destination = 4. The source is in stage 0 and destination is in stage 1, therefore Step 5 is used. Here the difference is calculated as diff = (4)mod 4 – (1)mod 4, which comes out to be -1. The tag can not be generated in this case as the connectivity between stage 0 and 1 is done using 20 connection pattern. The (desination)mod 4 > (source) mod 4, so the vertical link is chosen to go upwards. The SE 0 is reached and it will act as new source. Then again the tag is generated, which comes out to be 0, means using straight link in forward direction the destination can be reached. Figure 10 shows this routing.

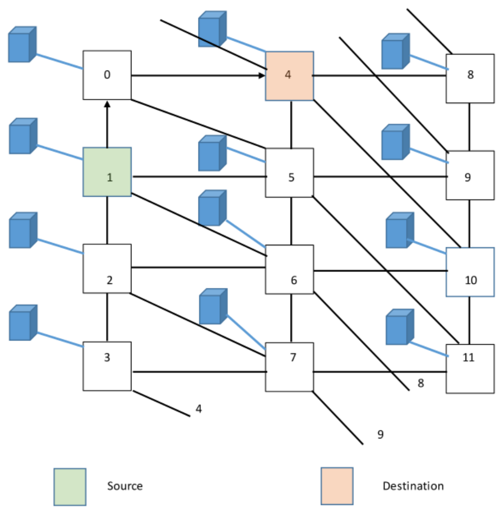

Fig 10: Routing packet from 1 to 4 using Step 5

Example 6: The source = 5 and destination = 3. The source is in stage 1 and destination is in stage 0, therefore Step 6 is used. Here the difference is calculated as diff = (5)mod 4 – (3)mod 4, which comes out to be -2. The tag can not be generated in this case as the connectivity between stage 0 and 1 is done using 20 connection pattern. The (desination)mod 4 < (source) mod 4, so the vertical link is chosen to go downwards. The SE 6 is reached and it will act as new source. Then again the tag is generated, which comes out to be -1. Still due to connection pattern between stage 0 and 1 it is not possible to route the packet. So again the vertical link in doward direction is used , to reach SE 7. Now from SE 7 the difference becomes 0, means using straight link in backward direction the destination can be reached. Figure 11 shows this routing.

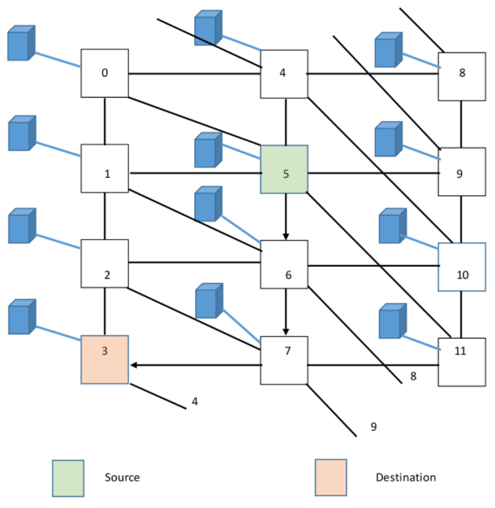

Fig 11: Routing packet from 5 to 3 using Step 6

3. RESULTS AND CONCLUSION

In this section we present the experimental setup, results and discussion. The simulation of this algorithm is written in C++. The simulation is written for fault-free environment. It is also assumed that each SE has sufficient buffer space to hold the packets to be routed. The simulation has been tested on Dell Vostro Quad Core processor with 64 bit Ubuntu OS. The input output pairs are generated randomly during this testing. It was observed that in hop count has not exceeded 4, which is an achievement of this work. The algorithm can easily be generalized for any size of NoCGIN.

4. FUTURE SCOPE

In this paper, a new variant of GIN is proposed, whose name is NoCGIN. As the name suggests, this variant is very useful as NoC interconnect. The prominent observations are:

1. The network allows forward as well as backword routing, which was not available in original network,
2. Each SE can be connected with one core, makes it suitable for NoCs with high number of cores,
3. The routing algorithm is very simple and can be easily genralized.

As it is mentioned in Section 3, the simulation is tested in fault-free environment. The same algorithm can be extended to work in faulty environment. The hop count which was observed not beyond 4 will certainly change in that scenario. Further we would also wish to try the use of other GIN variants in NoC, to explore their usability.

5. REFERENCES

[1] Hwang, K., and Briggs, F.A., 1984, Computer Architecture and Parallel Processing, McGraw– Hill, New York.

[2] Dally, W., and Towles, B., 2004, Principles and Practices of Interconnection Networks, Morgan Kaufmann, San Francisco, CA.

[3] Duato, J., Yalamanchili, S., and Ni, L.M., 2003, Interconnection Networks: An Engineering Approach, Morgan Kaufmann, and San Francisco, CA.

[4] Feng, T. Y., 1981, “A Survey of Interconnection Networks”, IEEE Transactions on Computers.

[5] Adams III, G. B., Agrawal, D. P., and Siegel, H. J., 1987, “A Survey and Comparison of Fault–Tolerant Multistage Interconnection Networks”, IEEE Transactions on Computers.

[6] Special Issue on Interconnection Networks, 1987, IEEE Computer 20 (6).

[7] Siegel, H.J., 1990, Interconnection Network for Large Scale Parallel Processing: Theory and Case Studies, McGraw Hill.

[8] Nitin, 2002, On a fault–tolerant hybrid ZETA MIN, Master’s Thesis, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab.

[9] Nitin, 2006, “Component Level Reliability Analysis of Fault–tolerant Hybrid MINs”, WSEAS Transactions on Computers.

[10] Kruskal, C. P., and Snir, M., 1983, “The Performance of Multistage Interconnection Networks for Multiprocessors”, IEEE Transactions on Computers, Vol. C-32, Issue 12.

[11] Raghavendra, C. S., and Varma, A., 1986, “Fault Tolerant Multiprocessors with Redundant-Path Interconnection Networks”, IEEE Transactions on Computers, Vol. C-35, Issue 4.

[12] Varma, A., and Raghavendra, C. S., 1989, "Fault-Tolerant routing in Multistage Interconnection Networks", IEEE Transactions on Computers, Vol. 38, Issue 3.

[13] Fan, C. C., and Bruck, J., 2000, "Tolerating multiple faults in Multistage Interconnection Networks with minimal extra stages", IEEE Transactions on Computers, Vol. 49, Issue 9.

[14] Parker, D. S., and Raghavendra, C. S., 1982, "The Gamma Network: A Multiprocessor Interconnection Network With Redundant Paths", IEEE Transactions on Computers.

[15] Parker, D. S., and Raghavendra, C. S., 1984, "The Gamma Network", IEEE Transactions on Computers, Vol. c–33, No. 4.

[16] Kothari, S. C., Prabhu G. M., and Roberts, R., 1988, "The Kappa Network with Fault–Tolerant Destination Tag Algorithm", IEEE Transactions On Computers, Vol.37 No 5.

[17] Lee, K.Y., and Hegazy, W., 1988, "The Extra Stage Gamma Network", IEEE Transactions on Computer, Vol. 37, No. 11.

[18] Lee, K. Y., and Yoon, H., 1990, "The B–Network: A Multistage Interconnection Network With Backward Links", IEEE Transactions on Computer, Vol. 39, No. 7.

[19] Venkatesan, R., and Mouftah, H. T., 1992, "Balanced Gamma Network–A New Candidate For Broadband Packet Switch Architectures", IEEE Transactions on Computer.

[20] Chen, C. W., Lu, N. P., Chen, T. F., and Chung, C. P., 2000, "Fault Tolerant Gamma Interconnection Networks By Chaining", IEE Proceedings – Comput. Digit. Tech, Vol. 147, No. 2.

[21] Chuang, P. J., 1998, "Creating a Highly Reliable Modified Gamma Interconnection Network Using a Balance Approach", IEE Proceedings – Comput. Digit. Tech, Vol. 145, No. 1.

[22] Tzeng, N. F., Chuang, P. J., and Wu, C. H., 1993, "Creating Disjoint Paths In Gamma Interconnection Networks", IEEE Transactions on Computer, Vol. 42, No. 10.

[23] Chuang, P. J., 1994, "CGIN: A Modified Gamma Interconnection Network with Multiple Disjoint Paths", IEEE Transactions on Computer.

[24] Chen, C. W., Lu, N. P., and Chung, C. P., 2003, "3–Disjoint Gamma Interconnection Network", The Journal of Systems and Software.

[25] Chen, Z., "A Class of Incomplete Gamma Interconnection Network", Available at: www.researchgate.com

[26] Borkar, M.A., 2010, "A Survey of Fault Tolerance Techniques Used in GIN", in National Conference EEC, 2010

[27] Borkar, M.A., and Nitin, 2011, "3D–CGIN: A 3 Disjoint Paths CGIN with Alternate Source", in Proceedings of ACC.

[28] Barlik, P. K., 2011, "FIR Filter IC Design Using Redundant Binary Number Systems", M.Tech Thesis, NIT Rourkela, India.

[29] Borkar, M.A., 2011, "3D–CGIN: A 3Disjoint Paths CGIN with Alternate Source", M.Tech Dissertation, UTU Dehradun, India.

[30] Borkar, M.A., and Nitin, 2012, "Network Status Aware Routing in 3D–CGIN", in Proceedings of ICCCS.

[31] Wu, Y., Liu, L., and Wang, Z., 1993, "Modified gamma network and its optical implementation", Journal of Applied Optics, Vol. 32, Issue 35.

[32] Chaoyang, C. Q., "A minimal cost dynamic rerouting gamma network", Available At: http://wr.cyut.edu.tw

[33] Rajkumar, S., and Goyal, N.K., 2014, "Design of 4–disjoint gamma interconnection layouts and reliability analysis of gamma interconnection networks", Journal of Supercomputing.

[34] Chen, C. W., and Chung, C. P., 2001, "Fault Tolerant Gamma Interconnection Networks without backtracking", Journal of Systems and Software.

[35] Borkar, M.A., Nitin, and Kumar, A., 2015, "A Survey on the Family of Gamma Interconnection Network", International Journal of Applied Engineering Research, Vol. 10, No. 24.

[36] Bell, S., Edwards, B., Amann, J., Conlin, R., et al., 2008, "TILE64 processor: A 64-core SoC with Mesh interconnect", In Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC'08).

[37] Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., et al., 2007, "An 80-tile 1.28 tflops network-on-chip in 65nm cmos", In Digest of Technical Papers of the IEEE International Solid-State Circuits Conference (ISSCC'07).

[38] Hoskote, Y., Vangal, S., Singh, A., Borkar, N., et al., 2007, "A 5-ghz mesh interconnect for a tera flops processor", IEEE Micro.

[39] Radetzki, M., Feng, C. C., Zhao, X. Q., and Jantsch, A., 2013, "Methods for Fault Tolerance in Network-on-Chip", ACM Computing Surveys.

[40] Tilera Announces the world's first 100-core processor with the new tile-gx family, Available At: http://goo.gl/K9c85

[41] Nychis, G., Fallin, C., Moscibroda, T., Mutlu, O., and Seshan, S., 2012, "On-chip Networks from a Networking Perspective: Congestion and Scalability in Multi-Core Interconnects", In SIGCOMM.

[42] University of Glasgow, "Scientists squeeze more than 1,000 cores on to computer chip.", Available At: http://goo.gl/KdBbW

[43] Nitin, 2012, "On Asymptotic Analysis of Packet and Wormhole Switched Routing Algorithm for Application-specific Network-on-Chip", Journal of Electrical and Computer Engineering.

[44] Nitin, and Chauhan, D.S., 2010, "Stochastic Communication for Application Specific Networks-on-Chip", Journal of Supercomputing, Springer, Volume 59, Number 2.

[45] Agarwal, A., Iskander, C., and Shankar, R., 2009, “Survey on Network on Chip (NoC) Architectures & Contributions”, Journal of Engineering, Computing and Architecture, Vol. 3, Issue 1.

[46] Constantinescu, C., 2003, “Trends and challenges in vlsi circuit reliability”, IEEE Micro.

[47] Chae-Eun, R., Han-You, J., and Soonhoi, H., 2004, “Many-to-many core-switch mapping in 2-D mesh NoC architectures”, Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors.

[48] Nitin, 2006, “Component Level reliability analysis of fault-tolerant hybrid MINs,” WSEAS Transactions on Computers, vol. 5, no. 9.

[49] Nitin, and Subramanian, A., 2008, “Efficient algorithms and methods to solve dynamic MINs stability problem using stable matching with complete ties,” Journal of Discrete Algorithms, vol. 6, no. 3.

[50] Bjerregaard, T., and Mahadevan, S., 2006, “A survey of research and practices of network-on-chip,” ACM Computing Surveys, vol. 38, no. 1.

[51] Pirretti, M., Link, G. M., Brooks, R. R., Vijaykrishnan, N., et al, 2004, “Fault Tolerant algorithms for network-on-chip interconnect”, Proceedings of IEEE Computer Society Annual Symposium on VLSI.

[52] Holsmark, R., and Kumar, S., 2005, “Design issues and performance evaluation of mesh NoC with regions”, Proceedings of 23rd NORCHIP conference.

[53] Rehan, F., Alemzadeh, H., Safari, S., Prinetto, P., et al, 2008, “Relaibility in Application Specific Mesh-based NoC Architectures”, Proceedings of 14th IEEE International Online Testing Symposium.

[54] Holsmark, R., Palesi, M., and Kumar, S., 2008, “Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions”, Journal of System Architecture, Vol. 54, Issue 3-4.

[55] Samuelsson, H., and Kumar, S., 2004, “Ring road NoC architecture”, Proceedings of NORCHIP conference.

[56] Bononi, L., and Concer, N., 2006, “Simulation and analysis of network on chip architectures: ring, spidergon and 2D mesh”, Proceedings of the Conference on Design, Automation and Test in Europe.

[57] Bononi, L., Concer, N., Grammatikakis, M., Coppola M., et al, 2007, “NoC Topologies Exploration based on Mapping and Simulation Models”, Proceedings of 10th Euromicro conference on Digital System Design Architectures, Methods and Tools.

[58] Manevich, R., Walter, I., Cidon, I., and Kolodny, A., 2009, “Best of both worlds: A bus enhanced NoC (BENoC)”, 3rd ACM/IEEE Symposium on Networks-on-Chip.

[59] Thid, R., Sander, I., and Jantsch, A., 2006, “Flexible Bus and NoC Performance Analysis with Configurable Synthetic Workloads”, Proceedings of 9th Euromicro conference on Digital System Design Architectures, Methods and Tools.

[60] Lee, H. G., Chang, N., Ogras, U. Y., and Marculescu, R., 2007, “On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches”, ACM Transactions on Design Automation of Electronic Systems, Vol 12, Issue 3.

[61] Tsai, K. L., Lai, F., Pan, C. Y., Xiao, D. S., et al, 2010, “Design of low latency on-chip communication based on hybrid NoC architecture”, Proceedings of 8th IEEE conference on NEWCAS.

[62] Mirza-Aghatabar, M., Koohi, S., Hessabi, S., and Pedram, M., 2007, “An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models”, Proceedings of 10th Euromicro conference on Digital System Design Architectures, Methods and Tools.

[63] Concatto, C., Almeida, P., Kastensmidt, M., Cota, E., et al, 2009, “Improving yield of torus nocs through fault-diagnosis-and-repair of interconnect faults”, Proceedings of 15th IEEE International Online Testing Symposium.

[64] Kao, Y. H., Alfaraj, N., Yang, M., and Chao, H. J., 2010, “Design of High-Radix Clos Network-on-Chip”, 4th ACM/IEEE International Symposium on Networks-on-Chip.

[65] Kao, Y. H., Yang, M., Artan, N.S., and Chao, H. J., 2011, “CNoC: High-Radix Clos Network-on-Chip”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 30, Issue 12.

[66] Pande, P. P., Grecu, C., Jones, M., Ivanov, A., et al, 2005, “Performance evaluation and design trade-offs for network-on-chip interconnect architectures”, IEEE Transactions on Computers, Vol. 54, Issue 8.

[67] Zeferino, C. A., and Susin, A. A., 2003, “SoCIN: a parametric and scalable network-on-chip”, Proceeding of 16th Symposium of Integrated Circuits and Systems Design.

[68] Ogras, U. Y., Hu, J., and Marculescu, R., 2005, “Key research problems in NoC design: a holistic perspective”, Proceedings of 3rd IEEE/ACM/IFIP International Conference on Hardware/Software codesign and System synthesis.