International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-1 Issue-5, June 2012

Reliable Routing & Deadlock free massive NoC Design with Fault Tolerance based on combinatorial application.

Deepthi chamkur .V , Vijayakumar.T

Abstract*—Technological evolution enables the integration of billions of transistors on a chip. As VLSI technology scales, and processing power continues to improve, inter-processor communication becomes a performance bottleneck. On-chip networks have been widely proposed as the interconnect fabric for high performance SoCs. Recently, NoC architectures are emerging as the candidate for highly scalable, reliable, and modular on-chip communication infrastructure platform. This paper proposes the generalized binary de Bruijn (GBDB) graph based on combinatorial application as a reliable and efficient network topology for a large NoC. We propose a deadlock free & reliable routing algorithm to detour a faulty channel between two adjacent switches. In this implementation, using just two-layer VLSI layout, we can implement a NoC with any desired number of nodes. Note that current VLSI technology allows more than two wiring layers and the number is expected to rise in the future. Our experimental results show that the latency and energy consumption of the generalized de Bruijn graph are much less than those of Mesh and Torus. The low energy consumption of a de Bruijn graph-based NoC makes it suitable for portable devices which have to operate on limited batteries. Also, the gate level implementation of the proposed reliable routing shows small area, power, and timing overheads due to the proposed reliable routing algorithm.

***Index Terms*— Network on chip (NoC), combinatorial application, energy consumption.**

I. INTRODUCTION

The need for scaling down of transistor can be attributed to the massive growth in the demand of electronics products all over the globe. And, scaling has given industry the power to ram more functionality in their products in the same or less chip area. Advances in VLSI technology will soon allow a single chip to contain more than one billion transistors, indicating that a large number of processing units (such as CPU, DSP, multimedia processor) shall be integrated into one packaged chip. Until now, we have witnessed the technology shift from large and robust computing system to System-on-Chip (SoC's). But, to meet the existing high bandwidth computing requirements, SoC's has to integrate more number of IP blocks. Traditionally, the communication between these processing elements was based on buses. However, this bus oriented structure has limitations & it is expected that the BUS will become a bottleneck from a performance, scalability and power dissipation point of view**.** And will not be able to support the heavy communication traffic of the current applications**.** Especially, when the current trend of implementing more cores on a single chip is starting to be widely adopted by manufacturers, the problem of communication between many processing cores will be a major area of concern. The scalability and success of switch-based networks and packet based communication in parallel computing and Internet has inspired the researchers to propose the Network-on-Chip (NoC) architecture as a viable solution.

With a communication-centric design style, Networks-on-Chip (NoC) was proposed to mitigate the complex communication problems. A NoC system is composed of a large number of processing units communicating to other units through routers across the interconnection network & provides a scalable infrastructure to interconnect different IPs and sub-systems in a SoC. Moreover, NoC's can make SoC's more structured, reusable, and can also improve their performance. It enables integration of a large number of computational and storage blocks on a single chip.Since the communication between the various processing cores will be deciding factor for the performance of such systems, therefore we need to focus on making this communication faster but at the same time more reliable also.

As the number of the elements and transactions among cores of the Multiprocessor SoC (MPSoC) and Multi core processor, (MCP) increases, the reliability and performance of the system becomes a key design and implementation issue for large scale system. Fault either transient or permanent and packet congestion in the interconnection network are sources for reducing the reliability and performance in a NoC based system. Packets travel through the network by passing one or more hops between the source and the destination tiles. As semiconductor technology scales down to nanometer domain, processing units (PUs), routers (nodes), and interconnect links (links) are all subject to new types of malfunctions and failures that are harder to predict and avoid. Particularly, failures of nodes/links may isolate a large number of fault-free PUs. A major challenge in NoC designs thus is to provide fault tolerance under link/node failure(s). So, it is imperious for us to lay stress on fault tolerance of Network-on-Chips.

Manuscript published on 30 June 2012.
***** Correspondence Author (s)
Deepthi chamkur V*, Dept.., of Electronics & communication Engineering, SJBIT ,Visveshwarirh Technological University, Bangalore – 60, India,.
Mr.Vijayakumar.T, Asso., professor, Dept.., of Electronics & communication Engineering, SJBIT ,Visveshwarirh Technological University,Bangalore–60,India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

II. BACKGROUND

The scaling of MOS transistors into the nanometer regime opens the possibility for creating large scalable Network-on-Chip (NoC) architectures containing hundreds of integrated processing elements with on-chip communication. NoC architectures, with structured on-chip networks are emerging as a scalable and modular solution to global communications within large systems-on-chip. The basic concept is to replace today's shared buses with on-chip packet-switched interconnection networks.

Network-on-Chip is a communication network that is used on chip. It replaces dedicated design specific wires with scalable, general purpose multi hop network. NoC provides a robust, high performance, scalable and power efficient communication infrastructure for connecting MPSoC components. Network based parallel processing using commodity Components, such as personal computers, has received attention as an important parallel computing environment.

NoC architectures use layered protocols and packet-switched networks which consist of on-chip routers, links, and well defined network interfaces. The principal goals of packet switching are to optimize utilization of available link capacity and to increase the robustness of communication. This is the reason why most networks proposed in the literature are packet based. The NoC architecture basic building block is the "network tile". The tiles are connected to an on-chip network that routes packets between them. Each tile may consist of one or more compute cores and include logic responsible for routing and forwarding the packets, based on the routing policy of the network. The structured network wiring of such a NoC design gives well-controlled electrical parameters that simplifies timing and allows the use of high performance circuits to reduce latency and increase bandwidth. These tiled architectures show promise for greater integration, high performance, good scalability and potentially high energy efficiency.

Many researchers have proposed the NoC as a scalable communication fabric for MPSoCs and MCPs. Angiolini *et al.*analyze the strengths and weaknesses of NoCs by performing a thorough analysis based on actual chip floor plans after the interconnection place & route stages and after a clock tree has been distributed across the layout. Intel has announced an 80-core prototype to deliver more than one trillion floating point operations per second. In the elaborated chip, each core contains a five-port message passing switch as well as the computing element. Cores in this chip are connected in a 2-D mesh network that implements a message passing scheme. In addition to design issues, the great concern about reliability and performance of on-chip networks has prompted wide research in the literature. A group of fault-tolerant and high-performance interconnection designs try to bypass faulty channels. Yang *et al.* proposes two fault-tolerant routing schemes in the presence of either single link/node failure or multiple link/node failures. The proposed routing schemes are based on a deterministic routing. As of the single link/node failure case, they show that the number of switches on the detoured route generated by the proposed routing scheme is at most two more hops than the number of hops on the original route. Note that, this approach is only applicable on the mesh topology.

Many researchers have considered 2-D mesh topology to implement NoCs. While the design complexity of this 2-D topology is modest and the wiring is simple and regular, it suffers from several disadvantages such as a large network diameter, energy inefficiency because of the high hop counts, and low reliability. With the trend towards increasing the number of cores in MCPs and MPSoCs, the on-chip network needs to be scaled efficiently. Providing this scalability, researchers have studied some other topologies for NoC. Moussa, *et al.* proposed to use multistage interconnection networks, which are adapted Benes and Butterfly networks, as on-chip communication networks for parallel turbo decoding. Kim, *et al.* proposed the use of high-radix switches in on-chip networks and describes how the flattened butterfly topology can be mapped to on-chip networks. By using high-radix switches to reduce the diameter of the network, the flattened butterfly offers lower latency and energy consumption than conventional on-chip topologies. The network topology has a great impact on the system performance and reliability. There are well-defined relationships between the NoC topology and the packet delay, the routing algorithms, fault-tolerance, and fault-diagnosis.

The motivation behind this paper is to investigate new topologies for reliable and high-performance massive NoCs. The **de Bruijn** graph as a suitable topology for an on-chip network communication is considered in this paper. The degree of these graphs is bounded (i.e., the degree of the network does not change with an increase in the size of network). These graphs also have advantages such as *small diameter*, *high connectivity*, *easy routing*, and *high reliability*.

III. PRELIMINARIES

This paper proposes the generalized binary de Bruijn (GBDB) graph as the on-chip interconnection network to communicate between cores in a SoC design. This graph has a number of unique features that makes it suitable to implement a high-performance, low energy consumption, and more reliable NoC. In this section, we explain a few definitions that are used in the rest of this paper.

Network-on-Chip: NoC architecture consists of two main parts: switches and links. To implement a SoC by the NoC, each core in the SoC is connected to a switch through an interface. Using a network topology, links connect the switches together. Thereby, paths between switches (and consequently cores) are created.

Generalized de Bruijn Graph: Generalized binary de Bruijn graph prepares a high speed network to perform the communication among cores in a NoC. This graph can be defined as follows.

A generalized de Bruijn graph, GDB(d,n), has n nodes, where n can be any desired integer value. Further, Generalized de Bruijn graph GDB(d, n) is Hamiltonian. Each two nodes i and j are connected together if they satisfy one of the following equations:

© Copyright: All rights reserved.

(for d=2)

$i \equiv 2*j + r \pmod n$, $r=0$ or 1 (1)

$j \equiv 2*I + r \pmod n$, $r=0$ or 1 (2)

GDB (*2, n*) is called generalized binary de Bruijn graph.

A generalized binary de Bruijn graph for n=14 nodes is shown in figure 6. As seen, except four nodes all other nodes have four links. Nodes *0* and *13* have just two links, and Nodes *4* and *9* have three links.

The number of links for each node in a generalized de Bruin graph can be determined by the under mentioned theorem.

Theorem 1: All nodes in a generalized binary de Bruijn graph have four links, except the following nodes:

a) Only two Nodes 0 and n-1 in a GDB(2, n) have two links.

b) ***If n=3k*** *(k is an integer number) then Nodes n/3, 2n/3, (n- 3)/3, and (2n-3)/3 have three links.*

c) ***If n=3k+1*** *then Nodes 2(n-1)/3 and (n-1)/3 have three links.*

d) ***If n = 3k+2*** *then Nodes (n-2)/3 and (2n-1)/3 have three links.*

A generalized de Bruijn graph possesses many features which makes it a strong contender for implementation of reliable network. The most important feature is the logarithmic relationship between the diameter of a generalized de Bruijn graph and the number of nodes in the graph.

Theorem 2: The diameter of a generalized binary de Bruijn graph GBDB, which is defined as the maximum among the lengths of shortest paths between all possible pairs of nodes, is not greater than $[\log_2 n]$.

Existence of Hamiltonian cycle is another feature of the generalized de Bruijn graph. Availability of the Hamiltonian cycle turned out to be the necessary condition for existence of the fault-tolerant graph, without it we cannot get the symmetry group order which is the minimum necessary order for fault-tolerance.

Despite the many suitable features of this graph (i.e., de Bruijn), just a few implementations of this topology can be found in the industry. The two main drawbacks of the de Bruijn graph are as follows.

1) **Expandability**: To expand a de Bruijn graph of degree and diameter to the next graph with the same degree, nodes should be added. It means that this graph cannot easily be expanded. Samatham, *et al.* proposed a technique to expand a de Bruijn graph. Preserving advantages of de Bruijn graphs, Reddy, *et al.* proposed the generalized de Bruijn graph which can have any number of nodes. However, to expand this graph (i.e., adding a node to a constructed network) almost all links in the network should be changed.

2) **VLSI Layout**: VLSI implementation of a de Bruijn graph needs proposing clever algorithms. Samatham, *et al.* proposed an optimum VLSI layout for de Bruijn graph. Apart from the computer network, we do not need to expand a NoC when it is implemented on a chip. Therefore, expandability is not a concern in NoC implementation. Note that, both de Bruijn graph and generalized de Bruijn graph are scalable. It means that using the same switch architecture and network algorithms (e.g., routing), we can construct a de Bruijn graph with any desired number of nodes. Hence, we can use the generalized de Bruijn graph to implement NoCs.

Note that using a suitable network topology to implement a NoC for either multi-core processors or multiprocessor systems can facilitate the proposing of low latency and more reliable routing algorithms.

IV. OUR PROPOSED NOC DESIGN

In this project, we develop a new architecture template, called Network on chip (NOC), for future integrated telecommunication systems. NoC template provides vertical integration of physical and architectural levels in system design. In the NoC template, a chip consists of contiguous areas called regions, which are physically isolated from each other but have special mechanism for communication among each other. A region of NoC will be composed of computing resources in the form of processor cores and field programmable logic blocks, distributed storage resources, programmable I/O and all these resources interconnected by a switching fabric, allowing any resource to communicate with any other resource. The NoC is instantiated by deploying a set of these components to form a topology and by configuring them in terms of buffer depth, etc. Some NoCs rely on specific topological connectivity, such as octagon or ring, to simplify the control logic, while others allow for arbitrary connectivity, providing more flexible matching to the target application.

In order to design a NoC, we have to propose switch architecture, a topology to connect switches together, and a routing algorithm. In this section, we explain these elements.

A. Switch Structure

The backbone of the NoC consists of switches, whose main function is to route packets from sources to destinations. In order to implement our proposed NoC, we have designed a simple virtual channel switch. The same as switches in the wide-area networks, this switch consists of three parts: input/output (I/O) ports, a routing algorithm, and a crossbar switch.

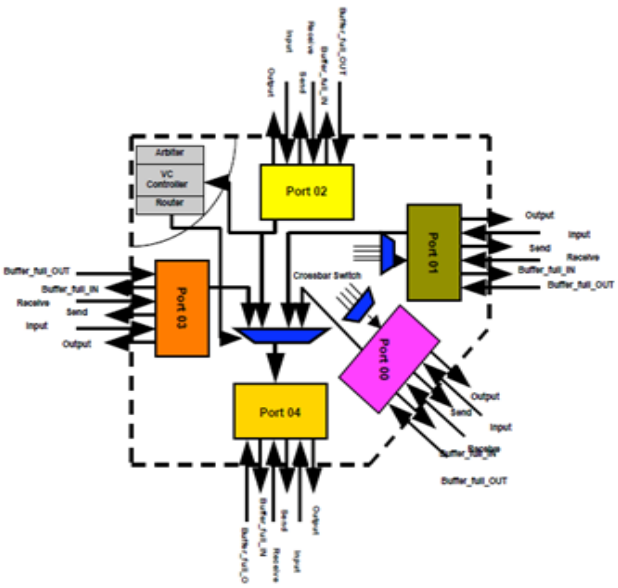

Fig.(1).Router Structure

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

The developed switch is completely structural, and consists of five ports each of which is processed independently from other ports and includes a multi channel circular FIFO. Each port is designed in such a way that it can send/receive data to/from other ports concurrently. In cases that two or more ports want to send data to a port, a round-robin arbiter hinders the collusion of data and grants to just one port. The granted port begins to send data, and other ports must wait until this port finishes with sending data. Fig. (1) shows the architecture of the proposed simple switch. Using the circular first-inputs first-outputs (FIFOs), each port implements a virtual channel scheme.

B. ***Topology***

The topology of the binary de Bruijn network is based on the directed de Bruijn graph. It is about a graph made up of $N=k^m$ nodes where each node is identified by a vector of size m in the base k. Connections between the nodes of the graph are done according to the following rule:

A node $V = V_m V_{m-1} \ldots V_1$ has an outgoing link to the node $U = U_m U_{m-1} \ldots U_1$ if and only if $V_i = U_i + 1$ with $i \in [1, m-1]$. The binary system having been considered in this work. Thus, the nodes of the graph are connected by unidirectional edges according to the rule: a node $V_m V_{m-1} \ldots V_1$ has its two output edges connected to the nodes $V_{m-1} V_{m-2} \ldots V_{10}$ & $V_{m-1} V_{m-2} \ldots V_{11}$.

Without loss of generality, we use the generalized de Bruijn graph with 14 nodes. Therefore, all the features, explained using this example, are applicable to other generalized binary de Bruijn graphs.

As shown in Fig. 2, using (1) and starting from Nodes 0 and n-1(i.e., Node 13), two link-disjoint binary spanning trees of generalized binary de Bruijn graph can be constructed. To construct the tree of Fig. 2(b), we start from Node 0 and using the modular equation (1), we obtain the connected nodes to Node 0 (i.e., Node 0, Node 1). This equation shows that there is a self-loop around Node 0. For the sake of generality and simplicity, we keep this self-loop in the tree structure. Node 0 is a parent for Node 1. The remainder part in (1) (i.e.., r) is used as a label for the link connected to corresponding Nodes. Using (1) for Node 1, we can obtain its children (i.e., Nodes 2 and 3).

Note that in this tree (Tree 1 of Fig. 2(b)), the node number of a child node is greater than the node number of its parent. We construct this tree until all children nodes have a node number less than their parent's node number. Therefore, all leaf nodes in this tree have a number not less than n /2 (where n is the number of nodes in the corresponding de Bruijn graph). Tree 2 of Fig. 2(c) can be constructed by using the similar technique.

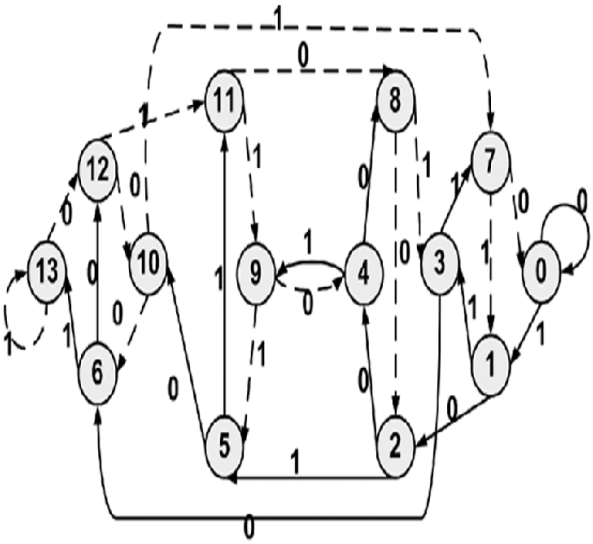

Fig. 2 (a). Two link-disjoint spanning trees for GBDB with 14 nodes

Note that there is a kind of analogy between Tree 1 and Tree 2. Each Node i in one tree (e.g., Tree 1) is corresponding to Node (n-i-1) in the other tree (e.g., Tree 2). For example two nodes i=2 and j=5 are connected together in Tree 1, and their corresponding nodes that are Nodes 14-2-1 = 11 and 14-5-1 = 8 are also connected together in Tree 2.Whereas links of these two trees are disjoint, these two spanning trees cover all links and nodes in the corresponding generalized de Bruijn graph. Moreover, all leaf nodes in Tree-1 are non-leaf nodes in Tree-2 and vice versa. Consequently, leaf nodes in these trees cover all nodes in the corresponding de Bruijn graph. Note that if the number of nodes, n, is a power of two, these binary trees are complete; otherwise, they are incomplete such as trees of Fig. 2. Based on Fig. 2, each node has at most two children and at most two parents. A directed link that connects a node to its parent represents a *Parental* relation (P-relation); and a directed link that connects a node to its child represents a *filial* relation (F-relation).

For example, the link between Nodes 12 and 10 in Fig. 2(a) shows a P-relation for Node 10 and an F-relation for Node 12. Labels that are shown on each link in Fig. 2 are assigned to each port of switches in a NoC. Using these labels and the link relation (*Parental* or *Filial*), each switch can distinguish a specific output port to route an incoming packet. The address of each core (connected to a switch in the NoC) is selected based on the code shown below the corresponding leaf node in these trees. Because a node in a generalized binary de Bruijn graph is a leaf node in just one of these two spanning trees, the node has a unique code.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

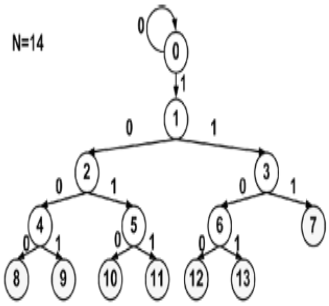

Fig. 2(b) Tree 1

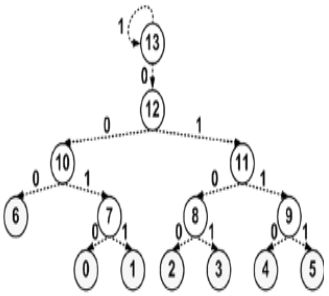

Fig. 2(c) Tree 2

Deadlock Free NoC Routing with Fault tolerance

This subsection explains how to find the shortest deadlock free path between two nodes in the GBDB. The diameter of a generalized de Bruijn graph has at most a logarithmic relationship with the number of nodes. Therefore, a simple routing algorithm can be implemented in switches of a NoC, with low area overhead.

Let us assume s and d be two nodes in GBDB(n) , and m be the diameter of the graph, then

$t = d - s\,2^m \pmod n$ (3)

is a binary number which defines ($t_{m-1}, \ldots\ldots t_1, t_0$) the path from s to d with length of m , so that

$s = u_m \rightarrow u_{m-1} \rightarrow \ldots\ldots \rightarrow u_1 \rightarrow u_0 = d$ (4)

where for $u_i = 2 * u_{i+1} + t_i \pmod n$ for i = 0,……., m-1.

For example, t = 0100 for s = 3 and d = 10 in GBDB (14) of Fig. 2. Therefore, the corresponding path is

$$S = 3 \xrightarrow{0} 6 \xrightarrow{1} 13 \xrightarrow{0} 12 \xrightarrow{0} 10 = d$$

Using this idea, Fig.3 shows the algorithm for finding the shortest path from Node to Node. The routing algorithm uses packet format of wormhole switching as shown in Figure 4. The four fields of the head flit in this figure are: source address, destination address, tags, and flit-type. Flit-type bits determine the head, data, and tail flits. The tag bits that are grouped into two subfields (i.e., Tag-1 and Tag-2) determine the exact route from the source to destination nodes.

The network uses source routing algorithm, so the source core generates the tag bits which go through certain minor modifications at each intermediate node along the path to the destination. When a message arrives at node *s*, there are two possibilities. First, the message may have to be delivered to the core connected to *s*, so the switch at *s* must deliver the packet to that core. On the other hand, if the message is destined for another core, the *s* must forward the packet to one of its neighbour switches which is the next hop in the path. The selection of the Particular switch is based on the corresponding tag bits in the tag field of head flit. The routing algorithm is implemented by adding in each of the switch two fields, called Tag-1 & Tag-2, to the head flit of a packet. These tags indicate to each intermediate node the next neighbour in the shortest path. Each bit in the *Tag-1* field determines the *F*- or *P*-relation in a switch along the shortest path, and each bit in the Tag-2 subfield determines the code assigned to each link.

```
1  Route(s, d) {
2      P={s};
3      i=0
4      if (s != d)
5          for(i = 1; t > n^i; i++)
6              t=(d-s*2^i)(mod n)
7      u=s;
8      while (i>0) {
9          u=2*u+(t/2^(i-1));
10         P=P+{u};
11         i--;
12     }
13 return P;
14 }
```

Fig.(3).

Deterministic shortest path routing algorithm

Source address, Destination address, Tag-1, and Tag-2 subfields have the same length of *k* bits (*k* is the diameter of the generalized binary de Bruijn which for our example is log14=4). Therefore, the length of head flit is (*k+k+k+k+2*) = *4k+2* bits (which the two first terms are numbers of bits in source and destination addresses, respectively, the third and fourth terms are numbers of bits in Tag-1 and Tag-2 and the last term shows the two bits needed to determine the type of flits in each packet).

The reliability of a NoC will be significantly influenced by the reliability of the connection between two neighbouring switches. In this section, we propose a reliable routing algorithm to detour a faulty connection. This connection is called reliable-channel (R-channel) which is defined as follows

R-channel: An R-channel is a path that a packet should traverse from one switch to its neighbouring switch. It consists of the virtual channel of receiving switch, signals of the sending switch, and the links between two switches. Two receive and transmit channels are exist between each two connected switches.

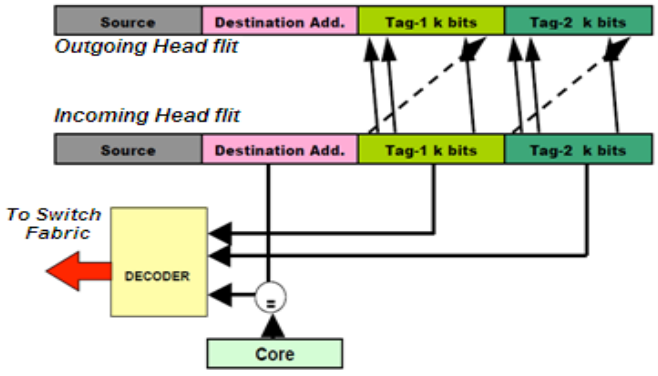

fig.(4) Routing Method

.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Faults on these R-channels may have the different erroneous effects like Packet Miss Routing, Packet Dropping and Data Error. In the data error case, although the packet is delivered to the desired destination, the data in the packet is not correct. If there is a fault in an R-channel, the routing algorithm should detour the faulty R-channel to deliver packets to the destination. In this section, we propose a reliable routing algorithm to detour the faulty R-channel for generalized de Bruijn topology with an even number of nodes. We assume that nodes are aware of the faulty R-channel that is connected to them by using a detection mechanism. In the sequel, we show that each-channel is in a loop with the length of no greater than four which is used to detour that -channel in the event of a fault. When the number of nodes in a generalized de Bruijn graph is even, child nodes of a parent node in one of the trees of Fig. 2 are the child nodes of another parent node in the other tree. The following lemma shows the relation between two parents of two children.

Lemma 1: In a generalized de Bruin graph with even nodes, two parents i (which i< n/2) and (n/2 + i) have the same children.

Fig. 5 shows the relations between parents and children that compose a *family*. Based on Lemma 1 the difference between node numbers of two parents of a family is n/2 and based on (1) and (2) the difference between node numbers of two children of a family is 1. Note that two Nodes 0 and 1 in Tree-2 of Fig. 2(c) have the same parent, that is Node 7, and the other parent is Node 0 in Tree-1 of Fig. 2(b). Also, Note that two Nodes 6 and 7 in Tree-2 of Fig. 2(c) have the same parent, that is Node 10, and the other parent is Node 2 in Tree-1 of Fig. 2(b).

Theorem 4: Each R-channel of a generalized binary de Bruijn-based NoC with even number of nodes is in a loop of length no greater than four.

Using this theorem, in the presence of a faulty R–channel between two nodes (e.g., Nodes I and s of Fig. 5), one of the nodes connected to the faulty R-channel (e.g., Node i) can send the packet to the other node (i.e., Node s) connected to the faulty channel, through other channel and nodes in the loop (e.g., and). Therefore, to detour a faulty R-channel, the packet should pass two extra links or two extra nodes.

Example 3: : In the previous example in which a packet is sent from Node 0 to Node 13, if the R-channel connecting Node 10 to Node 7 is faulty, the new path will be "0→ 7→ 3→ 6→ 10→ 6→ 13."

Each port in a switch has a status bit, f, that determines the status of the R-channel connected to that port. If this bit is "1," the R-channel is faulty; otherwise, the link is functional. A diagnostic circuit periodically checks the R-channel and asserts the corresponding bit, f, in the presence of a fault in the R-channel. Using the status bit (i.e., f) and bits in the tag fields of the head flit, each switch can detour a faulty R-channel using another R-channel connected to a member of its family, as shown in Fig. 5.

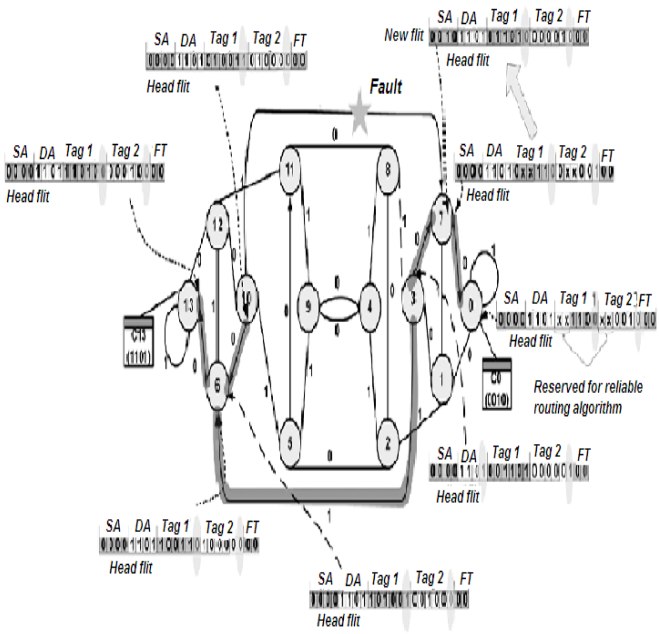

Fig.(5).Head flit from Node 0 to Node 13 in GBDB(14) when link between Nodes 7 and 10 is faulty.

Replacing the faulty R-channel with another R-channel in the family is a local decision that each switch can handle it easily. Using the family members to detour the faulty R-channel needs to add two extra hops to the original path. As a result, two extra bits in each tag are enough to detour a faulty R-channel. Therefore, in the fault-tolerant switch, we add two bits to each tag field, i.e., a tag field has $[\log_2 n] + 2$ bits. In a switch which changes the route, let us assume that the LSB of Tag-1 and Tag-2 are a and b respectively. If an R-channel through which the switch sends a packet is faulty (i.e., f = 1), by expanding the LSB of Tage-1 (i.e., a) to three bits, and expanding the LSB of Tage-2 (i.e., b) to three bits, the switch can detour the packet around the faulty R-channel using other members of the family. After this two-bit tag expanding, the switch connected to the faulty R-channel performs the normal routing algorithm. The following example elucidates the proposed reliable routing algorithm with more details.

Example 4: In Example 2 in which a packet traverses from Node 0 to Nodes 13, assume that the R-channel between Nodes 7 and 10 is faulty. The family covering R-channel from Node 7 to Node 10 consists of Nodes 7, 3, 6, and 10. Therefore, the new path from Node 0 to Node 13 is 0→ 7→ 3 6→ 10→ 6→ 13.Fig. 6 shows the modified head flit during this transfer.

V. VLSI IMPLEMENTATION

NoC implementation of multiprocessor systems requires the planarization of the interconnection network onto the silicon floorplan. Using VLSI technology with two wiring layers, the generalized de Bruijn graph can easily be implemented. Because a binary tree is a planar graph, we can implement the links in one of the trees in a wiring layer without intersection, and similarly, we can implement the links of the other tree in the second wiring layer.

© Copyright: All rights reserved.

Therefore, only two wiring layers are enough to implement a generalized binary de Bruijn graph with any number of nodes.

Regular tile-based implementation of NoC architecture was recently proposed to optimize the silicon area and delay between switches. As shown in Fig.6, such an implementation consists of a grid of regular tiles where each tile can be a core a switch, and channels to route the links that connect switches together. A switch is embedded within each tile with the objective of connecting it to the other tiles. Channels in a tile have some links that *bypass* the tile as well as some links that connect two switches in two adjacent tiles. The bypass links are used to connect two nonadjacent tiles. Our goal is to map the generalized binary de Bruijn topology to the tile-based architecture to minimize the use of bypass links in the implemented NoC. In the sequel, we formulate this problem and propose an integer linear programming (ILP) method to solve it.

Let us show a tile structure with p rows and q columns by T(p,q). Starting from left-top corner of the tile structure and moving to right, we assign a label from to t_o to t_{pq-1} tiles as shown in Fig. 6.Therefore, T = $\{t_0, t_1 \ldots t_{pq-1}\}$ is a set of tiles. We also use the set G = $\{g_0, g_1 \ldots g_{n-1}\}$ to show nodes in the generalized binary de Bruijn graph GBDB (n).The tile-based implementation is a mappingβ: G$\rightarrow$T, where $\beta(g_i) = t_j$ denotes that the switch g_i in the GBDB graph is mapped to the switch of tile t_j. Using a binary decision variable with two indices, X = $\{x_{ij}$; i=0,1…..,n-1;j = 0,1,2…pq-1;n $\leq$ pq$\}$, we can model the tile-based mapping, as follows:

$$Xij = \{1, if\ \beta(gi) = tj | 0,\ otherwise\} \quad (5)$$

The following constraints on the binary variables Xij should be considered for our model. First, each switch in the GBDB should be mapped to only one tile, that is

$$\sum_i \text{Xij} = 1 \quad (6)$$

Second, two different switches in the GBDB should not be mapped to the same tile, that is

$$\sum_j \text{Xij} \leq 1 \quad (7)$$

With these constraints we have to minimize the cost of interconnection between tiles. The total interconnection cost can be shown by (8), where Xij and Xkl show that switches I and k of the GBDB are mapped to tiles J and l , respectively; $\propto i_k$is a binary decision variable that is 1 only when two switches i and k in the GBDB are connected together, and C_{jl} is the cost of connecting tiles J and l

$$\sum \propto ik \text{ Xij Xkl Cjl.} \quad (8)$$

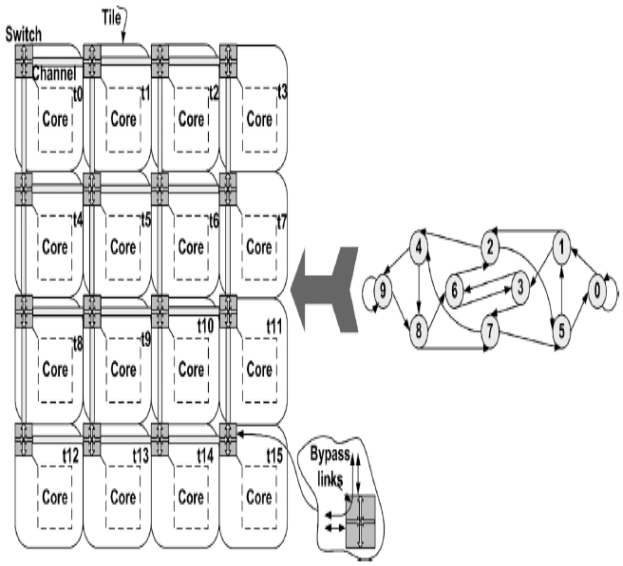

Fig (6) Tile-based generalized binary de Brujin topology

The cost C_{jl} is defined as the number of channels along the links connecting two tiles j and l in the tile architecture. Note that in tile architecture a channel is the connection between each two adjacent tiles. The cost function of (8) has a quadratic form. Because variables Xij and Xkl are binary, (8) can be converted to a linear form. Defining binary decision variables { Y_{ijkl}; i ,k = 0,1,…,n-1;j,l= 0,1,….,pq -1} to represent XijXkl, we can write the cost function as follows:

$$\sum_{i,j,k,l} \propto ik \text{ Yijkl Cjl.} \quad (9)$$

$$Y_{ijkl} \leq X_{ij} \quad (10)$$

$$Y_{ijkl} \leq X_{kl} \quad (11)$$

$$Y_{ijkl} > X_{ij} + X_{kl} - 1 \quad (12)$$

Equations (10) and (11) ensure that must be zero if either of X_{ij} or X_{kl}or are zero. (12) ensures Y_{ijkl} that is one if both X_{ij} and X_{kl} are one. The tile-based GBDB implementation can be formulated as follows:

Minimize $\sum_{i,j,k,l} \propto ik$ Yijkl Cjl.

Constraints: $Y_{ijkl} \leq X_{ij}$, $Y_{ijkl} \leq X_{kl}$,$Y_{ijkl} > X_{ij} + X_{kl} - 1$, $\sum_i$ Xij = 1 and $\sum_j$ Xij ≤ 1, where 0<i,k<n-1;0<j,l<pq-1; X_{ij} X_{kl},Y_{ijkl}. ϵ {0, 1}.

VI. EXPERIMENTAL RESULTS

To evaluate our techniques, we have implemented the proposed switch and reliability routing algorithm in RTL synthesizable Verilog. We have synthesized the switch with UMC 0.18-µm and VDD = 1.8 V technology using Synopsys design compiler tools. The rest of this section explains the results. The synthesizable switch has four different parts: input FIFO (with the size of five flits), switch fabric, arbiter, and switch (containing the virtual channel controller). Fig.7 shows dynamic power, leakage power, and area of different parts of a switch obtained by Synopsys design compiler.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Fig.7 (a) shows that FIFOs and switch parts consume about 88.34% of the total dynamic power consumption in a switch. In addition, 66.81% of the leakage current is consumed in FIFOs. The FIFOs also occupy 63.02% of the switch area. The design was simulated using ModelSim and was tested for functionality by giving various stimuli.

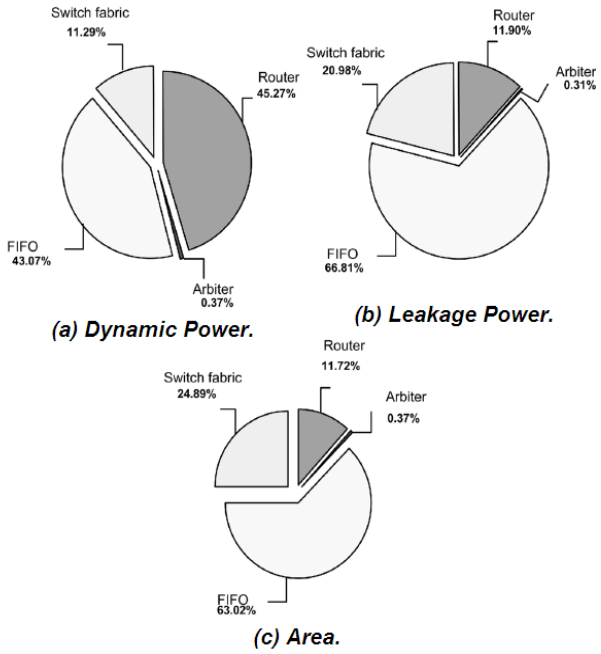

Fig.(7). Parameters of Different Switch Parts.

We have also synthesized the proposed switch structure for different sizes in three network topologies.Fig. 8(a) compares the area of switches with different sizes in three different topologies. As can be seen, the area of de Bruijn topology is much less than those of Mesh and Torus topologies. Note that the numbers within the figure are scaled by the area of the generalized de Bruijn graph with 100 nodes. As shown in this figure, the generalized de Bruijn graph based NoC is smaller than Mesh and Torus. Note that the size of a switch changes by the length of its flits. The length of the flit in a switch is determined by source address, destination address, and tag bits regarding to the topology and size of the network .The length of source/destination address has a logarithmic relationship with the size of topology. The tag length also has a direct relationship with the diameter of the topology.

For the proposed switch structure, the diameter has a dominant effect on the size of the switch area.Fig. 8(b), (c), and (d) show the dynamic power, leakage power, and propagation delay of each switch, respectively, in which switches in generalized binary de Bruijn topology show less dynamic and leakage power consumption as well as less delay in comparison with the switches of Mesh and Torus topologies.

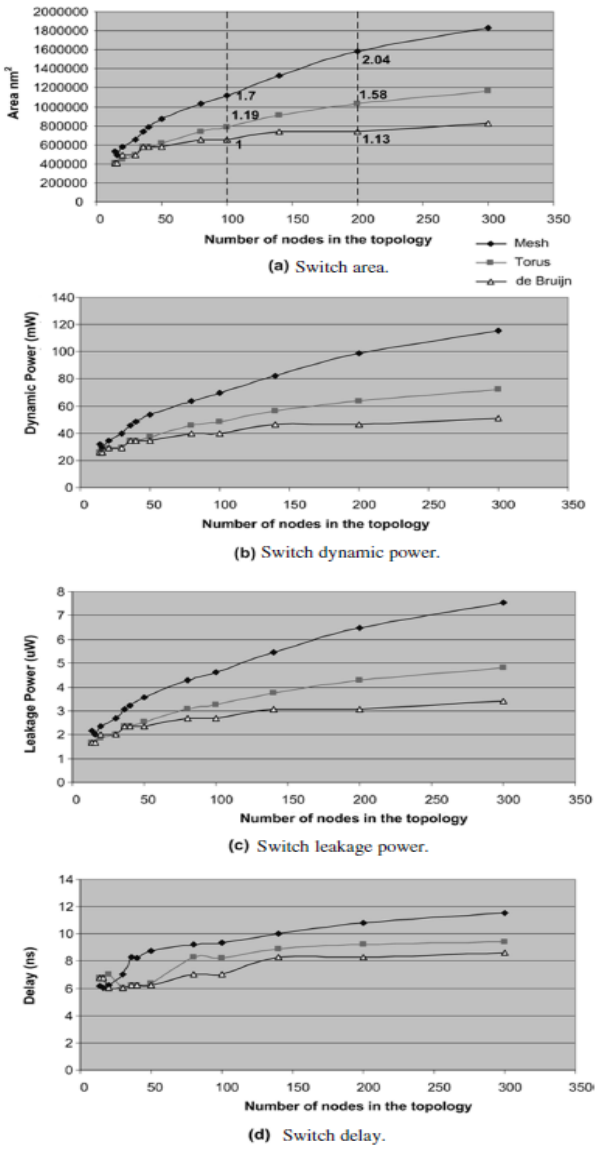

Fig.(8) SWITCH COMPARISION

Fig.9 compares the average packet latency for three different topologies. The latency is the time interval when a source node sends the first header flit until the destination receives the tail flit. To compute these average latencies, we have simulated various NoCs with different sizes and topologies. In this simulation, each switch sends different packets to all other switches. We have used dimension-order routing algorithms to deliver packets through shortest paths from sources to destinations in both Mesh and Torus networks.

Considering the power consumption and latency of the simulated traffic, we have computed the consumed energy which is shown in Fig. 10 after normalization. As can be seen, the GBDB is much suitable for energy efficient portable devices. To evaluate the efficiency of the proposed fault tolerant routing algorithm, we have injected a few faults in 20% of channels in GBDB-based NoCs with different sizes. Fig. 11 compares the faulty NoC with normal NoC in terms of dynamic power consumption.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

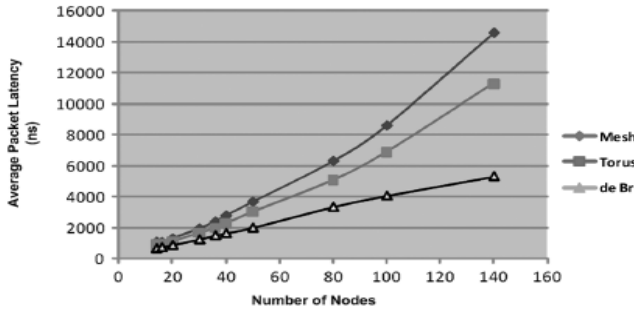

Fig.(9).Packet latency for three topologies

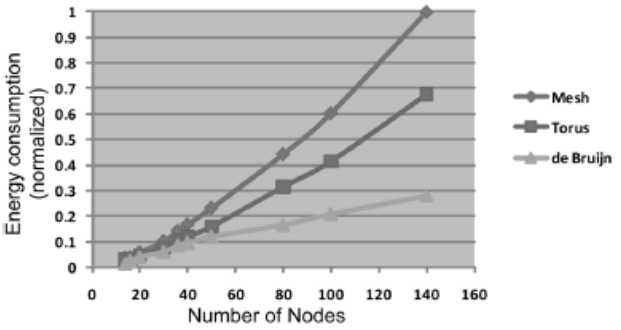

Fig.(10). Energy consumption for three topologies

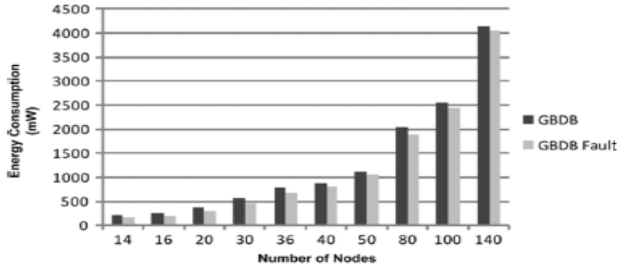

Fig.(11). Latency & Power overhead due to 20% of Faulty channels.

The proposed technique has only about 3.6% overhead on dynamic power consumption in large NoCs, respectively. We have used the lp solve, a general public software to solve the integer linear programming formulation.

Fig. 12 compares the number of tile channels needed to implement three different NoC topologies with different sizes on tiles. As can be seen, the number of channels needed by implementing a GBDB in a mesh of tiles is less than that of torus topology and is greater than that of mesh topology. Therefore, we expect the GBDB can run application faster than Mesh and Torus.

Tile dimension			# of used channel in the tile		
# of Row	# of column	# of nodes of NoC	mesh	Torus	GBDB
2	5	10	13	26	24
2	7	14	19	38	32
4	4	16	24	48	48
4	5	20	31	62	59
4	5	18	31	62	59
5	6	30	49	98	86

Fig.(12). Layout cost comparison.

This speed performance comes from two sources: the shortest paths in GBDB are much less than that of torus, and slight difference among the number of channels. Note that, we have considered the same clock frequency for the three NoCs. Considering different clock frequencies in different NoC topology is left for the future work which authors are working on it.

VII. CONCLUSION

This paper has investigated the use of generalized binary de Bruijn graph as the network topology in a large network-on chip.A deadlock-free reliable routing algorithm is presented considering channel faults. The experimental results show the efficiency of the proposed technique competed to Mesh and Torus topologies, in terms of latency, power, and area.

REFERENCES

[1] S. R. Vangal *et al.*, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[2] F. Angiolini, P. Meloni, S. M. Carta, L. Raffo, and L. Benini, "A layout aware analysis of networks-on-chip and traditional interconnects for MPSoCs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol.26, no. 3, pp. 421–434, Mar. 2007.

[3] M. Yang, T. Li, Y. Jiang, and Y. Yang, "Fault-tolerant routing schemes in RDT(2,2,1)/□-based interconnection network for networks-on-chip designs," in *Proc. 8th Int. Symp. Parallel Arch., Algorithms Netw.* (ISPAN), 2005, pp. 52–57.

[4] H. Moussa, O. Muller, A. Baghdadi, and M. Jezequel, "Butterfly and Benes-based on-chip communication networks for multiprocessor turbo decoding," in *Proc. Conf. Des., Autom. Test Eur. (DATE)*, 2007, pp. 654–659.

[5] J. Kim, J. Balfour, and W. J. Dally, "Flattened butterfly topology for on-chip networks," *IEEE Comput. Arch. Lett.*, vol. 6, no. 2, pp. 37–40, Jul. 2007.

[6] M. F. Karavai, "Minimized embedding of arbitrary Hamiltonian graphs in fault-tolerant graph and reconfiguration at faults," *Autom. Remote Control*, vol. 5, no. 12, pp. 2003–2019, 2004.

[7] M. R. Samatham and D. K. Pradhan, "The de Bruijn multiprocessor network:A versatile parallel processing and sorting network for VLSI,"*IEEE Trans. Comput.*, vol. 38, no. 4, pp. 567–581, Apr. 1989.

[8] D. K. Pradhan and S. M. Reddy, "A fault-tolerant communication architecture for distributed systems," *IEEE Trans. Comput.*, vol. C-31, no. 9, pp. 863–870, Sep. 1982.

Deepthi chamkur V pursuing M-tech in VLSI Design and Embedded Systems (Dept.., of Electronics & communication Engineering) in SJBIT, Bangalore – 60, Visveshwarirh Technological University, India,., and did my Engineering in Electronics and communication Engineering in GVIT, KGF, VTU and attended state level mathematical Talent exam and awarded 9th rank Area of Interest in Image processing, VLSI & Network on Chip .

Mr.Vijayakumar.T, working as a Asso., professor in Dept.., of Electronics & communication Engineering in SJBIT, Bangalore , *Having 15 years of experience as Faculty* , pursuing P.h.d in Kuvempu university . Area of Interest in Image processing, VLSI & Communication.

© Copyright: All rights reserved.