De Bruijn Graph as a Low Latency Scalable Architecture for Energy Efficient Massive NoCs

Mohammad Hosseinabady
University of Bristol, UK
mohammad@cs.bris.ac.uk

Mohammad Reza Kakoee
University of Tehran, Iran
kakoee@cad.ece.ut.ac.ir

Jimson Mathew
University of Bristol, UK
jimson@cs.bris.ac.uk

Dhiraj K. Pradhan
University of Bristol, UK
pradhan@cs.bris.ac.uk

Abstract

In this paper, we use the generalized binary de Bruijn (GBDB) graph as a scalable and efficient network topology for an on-chip communication network. Using just two-layer wiring, we propose an optimum tile-based implementation for a GBDB-based Network-on-Chip (NoC). Our experimental results show that the latency and energy consumption of generalized de Bruijn graph are much less with compared to Mesh and Torus, the two common NoC architectures in the literature.

1. Introduction

Multi-core processors (MCPs) and multiprocessor systems-on-chips (MPSoCs) have experienced a rapid development during the last decade. Network-on-Chip (NoC) is a promising technique for communication among cores in Multi-core processors (MCPs) and multiprocessor systems-on-chips (MPSoCs) designs. As the number of elements and transactions among cores in MCPs and MPSoCs increases, the power consumption and performance become a key issue in the design and implementation of large scale systems.

Many researchers have proposed the NoC as a scalable communication fabric for MPSoCs and MCPs. Angiolini et al. [1] analyze the strengths and weaknesses of NoCs by performing a thorough analysis based on actual chip floorplans 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 [2]. In this elaborate chip, each core contains a 5–port message passing router as well as the computing element. Cores in this chip are connected in a 2D mesh network that implements a message passing scheme. Hosseinabady et al. [3] propose the generalized de Bruijn graph as a reliable network topology. They also propose a reliable routing algorithm to detour a faulty switch.

The motivation behind this paper is to study the new topologies for high-performance massive NoCs as well as addressing their implementations. The **de Bruijn** graph (DB graph) as a suitable topology for an on-chip network communication is considered in this paper. This graph has many advantages such as *small and fixed diameter*, *high connectivity*, *easy routing,* and *high reliability*.

The DB($2, k$), which is called binary de Bruijn graph, can be obtained as follows. If we represent a node I by a k-bit binary number, say, $I = I_{k-1}I_{k-2}...I_1I_0$, then its neighbors can be represented as $I_{k-2}...I_1I_00$, $I_{k-2}...I_1I_01$, $0I_{k-1}I_{k-2}...I_1$, and $1I_{k-1}I_{k-2}...I_1$. In other words, this graph corresponds to the state graph of a shift register of length k. The shift register changes a state by shifting in a digit in the state number in one side, and then shifting out one bit digit from the other side.

The two main drawbacks of the de Bruijn graph are its expandability and its VLSI implementation. Based on these drawbacks, there are just a few implementations of this network in parallel processing field. Note that expandability is not a concern in the NoC implementation, because we do not need to expand an NoC when it is implemented on a single chip. The de Bruijn graphs 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.

Generalized binary de Bruijn (GBDB) graph, which is explained later, is used in this paper as the on-chip interconnection network to communicate between cores in an SoC design. We have proposed an optimum tile-based implementation for GBDB graph. In this implementation, using just two-layer wiring, we can implement the interconnection of an NoC with any desired number of nodes.

The rest of this paper is organized as follows: A few definitions are explained in the next section as preliminaries. Section 3 describes using of generalized de Bruijn graph in an NoC design and proposes a switch structure. Section 4 deals with the VLSI implementation of GBDB graph. Experimental results will be given in Section 5. Finally, Section 6 concludes the paper.

2. Preliminaries

A few definitions which are used in the rest of this paper are explained in this section.

2.1 Network-on-Chip

Switches and links are two main parts of an NoC architecture. Using a network topology, links connect the switches together. A network topology should allow each node to send packets to every other node. A routing algorithm determines the path along which a packet is delivered to the destination node. Source routing, which is considered in this paper, is a common routing algorithm in which the entire path from a source to a destination is known to the sender and is included when sending data.

978-3-9810801-3-1/DATE08 © 2008 EDAA

2.2 Generalized binary de Bruijn graph

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

Definition: A generalized binary de Bruijn graph, GDB(*2,n*) (or GBDB(*n*)), has *n* nodes, where *n* can be any desired integer value. Each two nodes *i* and *j* are connected together if they satisfy one of the following equations:

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

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

Example 1: Figure 1 shows the generalized binary de Bruijn graph for *n*=10 nodes. As it can be seen in this figure, each link in this graph connects two nodes whose node numbers satisfy one of the Equations (1) or (2). For example, Nodes *i*=8 and *j*=6 satisfy Equation (2) with *r=0,* that is $2*8+0 \equiv 6 \pmod{10}$.

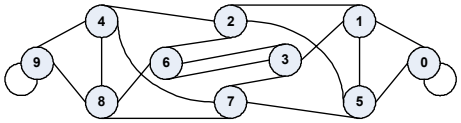

Figure 1 An example of GBDB(*2, 10*)

To use the same switch structure for all nodes in an NoC, we keep self-loops and redundant links between two nodes.

The generalized de Bruijn graph has a lot of features that make it suitable for implementation of reliable networks. The most important feature, which is denoted in Theorem 1, is the logarithmic relationship between the diameter of a generalized de Bruijn graph and the number of its nodes.

Theorem 1: The diameter of a generalized de Bruijn graph GDB(*d,n*), which is defined as the maximum among the lengths of shortest paths between all possible pairs of nodes, is not greater than $\lceil \log_d n \rceil$ [4].

3. NoC Architecture

Using a topology in an NoC, switches are connected together. Using this topology and a routing algorithm, switches route packets from a source core to destination cores.

In this section we explain a simple switch architecture as well as the NoC topology and its corresponding routing algorithm.

A simple virtual channel switch is used to implement our NoC [3]. We have considered a generalized binary de Bruijn graph as the network topology. Without loss of generality, we use the topology of Figure 1 as an example to explain our methods on this topology.

Using Equation (1) and starting from Nodes *0* and *n-1*(i.e., Node 9), we can fined two link-disjoint spanning trees which are shown in Figure 2.

To construct the tree of Figure 2-a, 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 binary complement of the remainder part in the Equation (1) (i.e., *r*) is used as a label for the link connected to corresponding Nodes. Using the Equation (1) for Node 1, we can obtain its children (i.e., Nodes 2 for r = 0 and 3 for r=1). Note that in this tree (Tree 1 of Figure 2-a), 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 parents' node number. Starting from Node 9 and using similar method, we can construct the spanning tree of Figure 2-b.

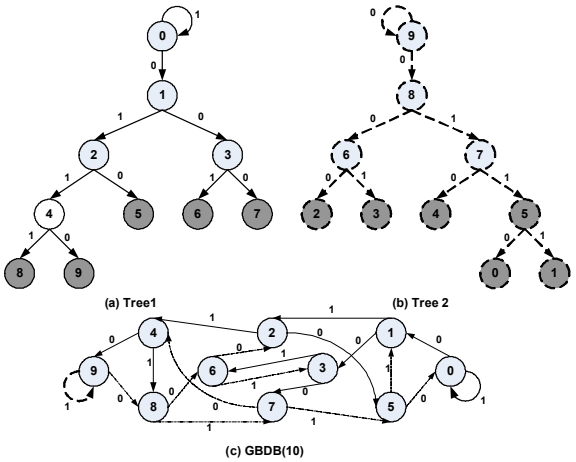

Figure 2 Two link-disjoint spanning trees for GBDB(14)

These two spanning trees cover all links and nodes in the corresponding generalized de Bruijn graph. Based on Figure 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 8 and 7 in Figure 2-b shows a *P*-relation for Node 8, and an *F*-relation for Node 7. Labels that are shown on each link in Figure 2 are assigned to each port of the switches in an NoC. Using these labels and the link relation (Parental or Filial), each switch can distinguish a specific output port to route an incoming packet [3].

4. VLSI implementation

NoC implementation of multiprocessor systems requires the planarization of the interconnect network onto the silicon floorplan. Using VLSI technology with two wiring layers, the generalized de Bruijn graph can easily be implemented. As we discussed in Figure 2, the two binary disjoint trees can cover all links in a generalized binary de Bruijn graph. Therefore, because a binary tree is a planar graph, we can implement the links of the threes in a wiring

layer without intersection, and similarly we can implement the links of the other tree in the second wiring layer.

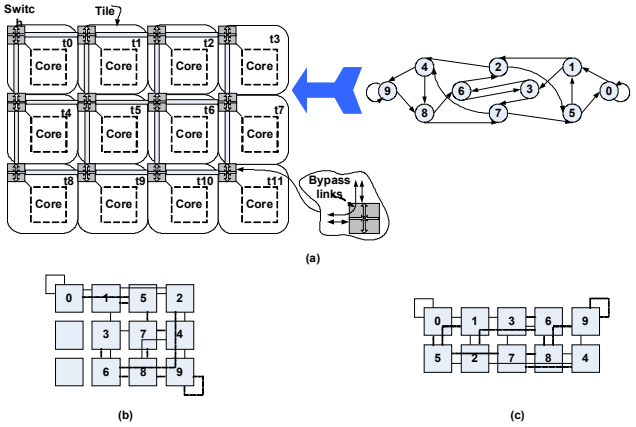

Figure 3 Tile-based generalized binary de Bruijn topology

Our goal is to map the generalized binary de Bruijn topology to a tile-based architecture (Figure 3) 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 to 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 t_0 to t_{pq-1} to tiles, like those of Figure 3-a. Therefore $T=\{t_0,t_1,t_2...,t_{pq-1}\}$ is a set containing tiles. We also use the set of $G=\{g_0,g_1,g_2...,g_{n-1}\}$ to show nodes in the generalized binary de Bruijn graph GBDB(n).

The tile-based implementation is a mapping $\beta: G \rightarrow T$, where $\beta(g_i)=t_j$ denotes that the switch g_i of the GBDB graph is mapped on the switch of tile t_j of T. Using a binary decision variable with two indices, $X=\{x_{ij}; i=0,1,2,...,n-1; j=0,1,2,..., pq-1; n \le pq\}$, we can model the tile-based mapping.

$$x_{ij}=\begin{cases}1 & if \quad \beta(g_i)=t_j \\ 0 & otherwise\end{cases} \qquad (3)$$

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

$$\sum_i x_{ij}=1 \qquad (4)$$

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

$$\sum_j x_{ij} \le 1 \qquad (5)$$

Considering these two constraints, we have to minimize the cost of interconnections between tiles. The total interconnection cost can be shown by Equation (6), where x_{ij} and x_{kl} show that switch i and k of the GBDB are mapped on tiles j and l, respectively; α_{ik} is a binary decision variable that is 1 only when the two switches i and k in the GBDB are connected together; and C_{jl} is the cost of connecting the two tiles j and l.

$$\sum \alpha_{ik} x_{ij} x_{kl} C_{jl} \qquad (6)$$

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 a tile architecture a channel is the connection between each two adjacent tiles. For example, in Figure 3-a the $C_{01}=1$ and $C_{24}=3$.

The cost function of Equation (6) has a quadratic form.

Example 5: Considering GBDB(10) and the tile architecture of size 2x5 and 3x4 and solving the corresponding integer linear programming problem, Figure 3-b, c show the mapping of GBDB on these tile architectures.

5. Experimental Results

We have implemented the proposed switch and reliability technique using RTL synthesizable VHDL code. We have also synthesized the switch with *UMC 0.18 µm* and *VDD=1.8v* technology using Synopsys Design Compiler tool. In this section, we discuss the obtained results.

The synthesizable switch has four different parts: input FIFOs, switch fabric, arbiter, and router (containing the virtual channel controller). Figure 4 shows dynamic power, leakage power, and area of different parts of the switch. As shown in this figure, FIFO and router parts consume about 88.34 percent of the all dynamic power consumption in the switch. In addition, 66.81 percent of the leakage power is consumed in FIFO parts. The FIFO parts also occupy 63.02 percent of the switch area.

We have also synthesized the proposed switch structure for different sizes of the three network topologies (i.e., Mesh, Torus, and Generalized binary de Bruijn). Figure 5 compares the area of switches with different sizes in three different topologies. As it can be seen in this figure, 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. For example for n=100, where n is the number of nodes, the Torus switch is 1.19 times and Mesh switch is 1.7 times larger than the switch in generalized de Bruijn topology; and for n=200, the Torus switch is 1.40 times and Mesh switch is 1.80 times larger than the switch in generalized de Bruijn topology.

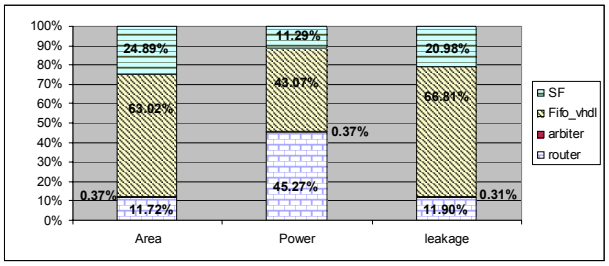

Figure 4 Parameters of different switch parts

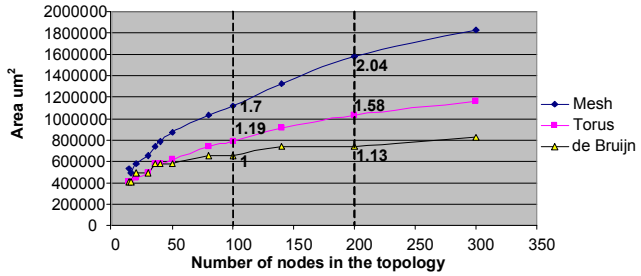

Figure 5 Switch comparison

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; also the tag length has a direct relationship with the diameter of the topology.

The same as the area, power consumption of the generalized binary de Bruijn switch is much less than that those of the two other topologies. In addition, the power consumption of a switch also has a linear relationship with the diameter of the topology in which the switch is used. Figure 6 compares the leakage power consumption of switches in the three different NoC topologies. It can easily be seen that the de Bruijn leakage power is much less than those of Mesh and Torus for large topologies.

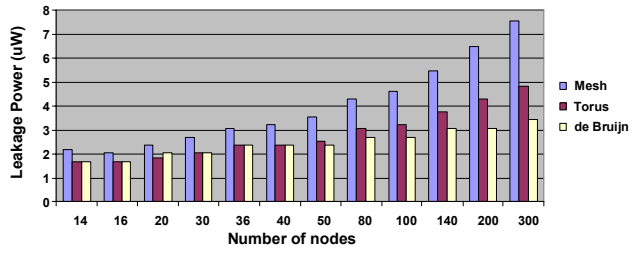

Figure 6 leakage power

Considering the power consumption and latency of the simulated traffic, we have computed the consumed energy which is shown in Figure 7 after normalization. As it can be seen, the GBDB is much suitable for energy efficient portable devices.

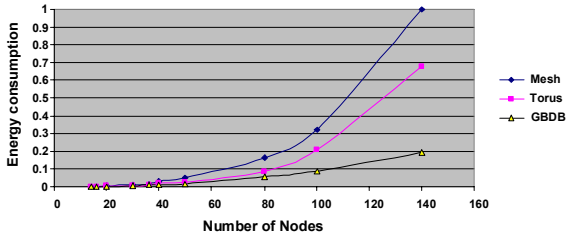

Figure 7 Energy consumption for three topologies

Using the proposed tile-based implementation of Section 5, Figure 8 compares the number of channels needed to implement the three different NoC topologies with different sizes on tiles.

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	56
5	6	30	49	98	86
6	6	36	60	120	98
5	8	40	67	134	126
10	5	50	85	170	140

Figure 8 Layout cost comparison

6. Conclusion

In this paper, we have used the generalized de Bruijn graph as the on-chip interconnection to communicate between cores in an SoC design. We have proposed an optimum tile based algorithm to implement the de Bruijn graph based NoC. Our experimental results show the efficiency of generalized de Bruijn graph to implement an NoC.

Acknowledgments

This work is funded by the Engineering and Physical Science Research Council (EPSRC). The authors would like to thank the reviewers of this paper for very insightful suggestions and comments for improving the paper.

References

[1] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 3, 2007.

[2] "Teraflops Research Chip," http http://www.intel.com/research/platform/terascale/teraflops.htm.

[3] M. Hosseinabady, M.R. Kakoee, J. Mathew, and D. K. Pradhan, "Reliable Network-on-Chip Based on Generalized de Bruijn Graph," *IEEE International High Level Design Validation and Test Workshop* (HLDVT'07), pp. 3-10, 2007.

[4] D. Z. Du, and F. K. Hwang, "Generalized de Bruijn digraphs," *Networks*, Vol. 18, pp. 27-38, 1988.