A Node Encoding of Torus Topology and Its Improved Routing Algorithm

Yang Xiaoqiang
School of Microelectronics, Xidian University, Xi'an, China
Email: xyangq@sina.com

Li Junmin, Du Huimin and Han Jungang
Department of computer science, Xi'an University of science and technology, Xi'an, China
Department of computer science, Xi'an Institute of Post & Telecommunications, Xi'an , China
School of Microelectronics, Xidian University, Xi'an, China
Email: yangxq581@163.com, {fv, hjg}@xiyou.edu.cn

***Abstract*—With the feature size of semiconductor technology reducing and intellectual properties (IP) cores increasing, on chip communication architectures have a great influence on the performance and area of System-on-Chip (SoC) design. Network-on-Chip (NoC) has been proposed as a promising solution to complex SoC communication problems and has been widely accepted by academe and industry. This paper discusses how to choose suitable topology and node encoding scheme for NoC, and proposes a two-dimensional plane code based on Johnson Code by the combination of Torus topology with corresponding node encoding. The node encoding implies the relation among neighbouring nodes and the global information of routing. And it has good scalable characteristics. The two methods for code compressing are also presented to reduce the storage space of node address and increase the utilization rate of channel bandwidth. Utilizing the code, the improved X-Y routing is proposed, which is implemented with only three or six logic operations in middle nodes. The node structure is designed at the same time. The experimental results show combination of the proposed code with Torus topology can simplify the routing algorithm in the implementation of NoC, decrease silicon resource consumption and greatly improve communication performance.**

***Index Terms*—Node Encoding, Network Topology, Routing Algorithm, Network on Chip, Compressed Code**

I. INTRODUCTION

With the development of VLSI technology, the realization of the future complex System-on-Chip (SoC) consisting of billions of transistors fabricated in technologies characterized by 65 nm feature size and less will soon be reality [1][2]. Such SoC implies the seamless integration of hundreds, or ever more, intellectual properties (IP) cores including processing resources and storage resources performing different functions and operating at different clock frequencies.

The most frequently used on-chip interconnect architecture is the shared medium arbitrated bus, where all communication devices share the same transmission medium. Every additional IP core connected to the bus adds to this parasitic capacitance, in turn causing increased propagation delay. As the bus length increases and the number of IP cores increases, the associated delay in bit transfer over the bus may will increase and eventually exceed the targeted clock period. This thus limits, in practice, the number of IP cores that can be connected to a bus and thereby limits the system scalability [3] [4]. With the increasing complexity of System on Chip, on chip communication architecture encounters some serious problems, such as power assumption, communication throughput and latency, clock synchronization etc.

Network on Chip (NoC) was introduced as an on chip communication facility and separates from computing functions [1]. The main techniques of NoC result from those of interconnection and communication of computer network and solve communication bottleneck problem of SoC at architecture level [2]. NoC takes full advantage of communication modes used in parallel computer systems and distributed computer systems, and utilizes the routing and packet switching instead of the on chip buses. It provides modular, scalable and reliable communication structure with high bandwidth. The main difference between NoC design and computer network is the limitation of chip area, power consumption and the physical effects in deep sub-micron and nano-micron technology [1] [3].

To implement the routing algorithms on NoC in VLSI technology, the key problems rest with simpler hardware structure with the limitation of chip resource and transmit latency [4] [5]. By analyzing the topology of Mesh, Hypercube and Torus and the characteristics of Gray Code and Johnson Code, the paper combines Torus topology with Johnson Code and presents two-dimensional plane code scheme which simplifies routing algorithms and hardware implementation and decreases the latency. The node encode implies the relation of

neighbouring nodes and their links. Utilizing the code, the improved algorithm for Y-X routing is present, which is implemented with only three or six logic operations in middle nodes.

The paper is organized as follows. Section II states related work, analyzes Mesh, Hypercube and Torus topology structure and the characteristics of Hypercube based on Gray Code. Section III introduces binary unit-distance cyclic code, Gray Code and Johnson Code and gives transformation between natural binary code and Gray Code and Johnson Code. The two-dimensional plane code based Johnson Code is proposed in Section IV. Section V puts forward two methods for compressed code. Section VI designs a kind of node structure for NoC and Section VII proposes the improved X-Y routing algorithm utilizing two-dimensional plane code based Johnson Code, which can decrease the cost of implementation. Section VIII analyzes performance followed by brief conclusions in section IX.

II. RELATED WORK

The topology of NoC concerns the placement and interconnection of NoC nodes. Protocols specify how these nodes and links work [1]. Several common topology of NoC are introduced in [1] [6]. Forms of topologies mainly includes regular and irregular forms. Mesh, Torus, Tree are regular forms of topologies shown as (a), (b), (c) in Figure 1. Irregular forms of topologies are derived by mixing different forms, and a hierarchical, hybrid or asymmetric fashion, as seen (c), (d) in Figure 1. Irregular forms of topologies scale non- linearly with regards to area and power, and are complicated in Wiring [7]. Of these NoC topologies, Mesh and Torus with regularity are mainly used in NoC design and prone to VLSI implementation [8] [9] [10].

Peter in [11] discussed Hypercube structure which is often utilized in parallel computer systems. In Hypercube structure, the node encoding based on Gray Code has one key property which is there exists a link between two nodes only if their binary addresses differ in a single bit [12]. This property is useful for deriving a number of parallel algorithms for the hypercube architecture and can simplify routing protocol implementation. The network performance of Mesh, Torus and Hypercube are compared in table I.

The cost of implementation of low-dimensional networks, such as Mesh, is lower than that of high-dimensional networks, such as Hypercube. In addition, Hypercube with dimension of five or six is not easy scalable and complicated in Wiring [12] [13]. Therefore, Hypercube is seldom used in NoC. Mesh is simple in hardware implementation and good in network scalability, so it is most widely used as topology for NoC [14]. Torus structure is an improvement of Mesh structure, which connects head node with tail node in each row and column to eliminate edge effect. Two-dimensional folded Torus presented by W.J.Dally from Standford University in [8] makes the distance between any two nodes equal and very node has same characteristics, so it is suitable for implementation in hardware.

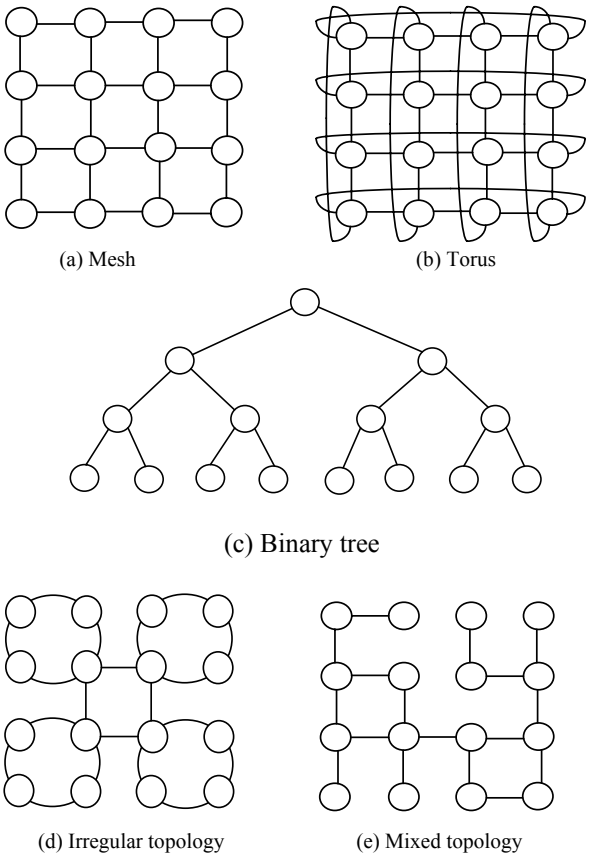

Figure 1. Three kinds of topology

The routing algorithms of NoC are divided into two categories by Ville Rantala in [10], deterministic routing and adaptive routing. The most elementary routing algorithm is X-Y routing. In X-Y routing, the packets follow the rows first then along the columns toward destination or vice versa [14] [15]. It is a deterministic routing algorithm of no deadlock and based on source, but its performance goes bad in heavy load. In [15] Jincao Hu presented an adaptive routing mode which avoids bad performance of X-Y Routing in heavy load. X-Y routing is used in light load or even load, and the adaptive routing is used in heavy load. It is called DyAD routing [15] [16]. Torus and Folded Torus have not only the excellences of Mesh which are simple in hardware implementation and good in network scalability, but also those of hypercube which can reduce the length of the shortest path and improve fault tolerant ability. However, the X-Y routing

TABLE I
COMPARING PERFORMANCE OF NETWORK (N NODES)

Topology	Mesh	Torus	Hypercube
Diameter	$2\left(\sqrt{N}-1\right)$	$2\lfloor\sqrt{N}/2\rfloor$	$\log_2\sqrt{N}$
Bisection width	$\sqrt{N}$	$2\sqrt{N}$	$N/2$
Arc connectivity	2	4	$\log_2\sqrt{N}$
Cost (link number)	$2\left(N-\sqrt{N}\right)$	2N	$(N\log_2)2$

and DyAD routing used in Torus are based on (x, y) coordinate encoding, their routing mechanism is more complicated than that of Hypercube using Gray Code. Our idea is to design NoC with Torus topology but using simple routing mechanism based on Gray Code in order to simplify routing hardware implementation.

III. BINARY UNIT-DISTANCE CYCLIC CODE

A. Binary Unit-distance Cyclic Code

Definition 1 A set of integral binary numbers [17] is called binary unit-distance cyclic code if

(1) any two neighbouring numbers have one and only one bit different (unit distance characteristic);

(2) the first number and the last one in the sequence have one and only one bit different (cyclic characteristic)

Definition 2 Given a set of natural numbers (0, 1,…, 2^k-2, 2^k-1), Each number of them is labeled binary form such as $A_{n-1}A_{n-2}\ldots A_1A_0$ and it can be mapped to $A_{n-1}(A_{n-1} \oplus A_{n-2})\ldots(A_3 \oplus A_2)(A_2 \oplus A_1)(A_1 \oplus A_0)$ which is called Gray Code, where " $\oplus$ " represents nonequivalence operation.

Definition 3 Given an increasing sequence (0, 1, 2… p-1, p, p+1 ,…, n-2, n-1), encode it with $m=\lceil n/2 \rceil$ bits, where n=2t, t=0, 1, 2,⋯. The encoded code satisfies the conditions of Definition 1, and if

(1) for p<m, the code of p is $Q=F_{m-1}\ldots F_kT_{k-1}\ldots T_0$, where k=p, "$F_{m-1}\ldots F_k$" is the sequence which consists of 0, while "$T_{k-1}\ldots T_0$" is the sequence which consists of 1, and if k=0, Q is the sequence which consists of 0.

(2) for p≥ m, the code of p is $Q=T_{m-1}\ldots T_k\ F_{k-1}\ldots F_0$, where k=p-m, "$F_{k-1}\ldots F_0$" is the sequence which consists of 0 while "$T_{m-1}\ldots T_k$" is the sequence which consists of 1, and if p=m, Q is the sequence which consists of 1. This code sequence is called Johnson code.

B. Transformation between NaturalBinary Code and Two Kinds Code

Algorithm 1 The transformation between natural binary code and Gray Code is such as the following.

(1) According to Definition 2, each natural binary code can be mapped to Gray Code .

(2) Each binary Gray Code $A_{n-1}A_{n-2}\ldots A_1A_0$ can also be mapped to a natural binary code $B_{n-1}B_{n-2}\ldots B_1B_0$, where $B_k=A_k \oplus B_{k+1}$, $0\le k<n-1$ and $B_{n-1}= A_{n-1}$.

Algorithm 2 The transformation between natural binary code and Johnson code is such as the following.

(1) According to Definition 3, each natural binary code can be mapped to Johnson code.

(2) Accordingly, a Johnson Code sequence（A_{n-1}, A_{n-2}, …, A_p,…, A_2, A_1, A_0）has corresponding natural binary sequence which is continuous and $m=\lceil \log_2 n \rceil$ bits in bit width as follows.

① If the highest bit of Ap is 0, its natural binary code is a binary data which is equal to the sum of the number of 1 in Ap;

② If the highest bit of Ap is "1", its natural binary code is binary data which equals the sum of the total number of 0 in Ap and m.

Both Gray Code and Johnson Code possess the unit distance characteristic and the cyclic characteristic, and are binary unit-distance cyclic code. In respective code set, the minimum distance between two codes is Haming Distance that is equal to the number of bits that are different in the two codes and is given by simple nonequivalence operation. But there are different characteristics and dissimilar applications between the two kinds of codes. 2^n Gray Codes can be composed of n bits while only 2n Johnson Codes can be done. The data encoded by Gray Code need smaller storage space than Johnson Code. Satisfied with cyclic and unit distance characters, Gray Code need to increase at least 2^n codes while Johnson Code 2 when their bit width increase 1bit. Therefore, Johnson Code is provided with better utilizing rate of space and its scalable characteristic is more flexible than Gray Code.

IV. TWO-DIMENSIONAL CODE BASED ON JOHNSON CODE

A. Two-dimensional Unit Cyclic Distance Code

Definition 4 When binary unit cyclic distance code is extended to two-dimensional plane in a manner, for each plane, two-dimensional code ought to possess two characters as follows.

(1) Each neighbouring node codes exist dissimilar one and only one bit; (unit distance characteristic)

(2) The difference between the first code and the last one in any row or column exist only one bit.(cyclic characteristic).

The code is called two-dimension unit cyclic distance.

The characters show binary unit distance cyclic code on both the row and column of two- dimensional plane satisfies the characters of one-dimensional code, and takes on some added traits:

(1) In the plane, it's a certainty that any node has four neighboring node (When node number is equal or greater than four)

(2) In the plane, all rows and columns exist one-dimensional unit distance cyclic ring.

Definition 5 Two nodes in two-dimension plane is named as adjacency, if there exists difference between their encodings, and only one bit varies.

Definition 6 If two arbitrary nodes in two-dimensional plane are adjacency then a (direct) link exists between them, according to Definition 4.

B. Two-dimensional Code Based on Johnson Code

Definition 7 The two-dimensional code based on Johnson Code (2-d JCode) is a kind of two-dimensional binary unit-distance cyclic code. Let 2-d Johnson Code is $w(w \ge 1, w \in z)$ bits in width. 2-d JCode=JCode_Y +JCode_X , and w=h+k, where JCode_X is Johnson Code with $h(h \ge 0, h \in z)$ bits width , JCode_Y is Johnson Code with $k(k \ge 1, k \in z)$ bits width and "+" denotes join operation, and if

(1) any two neighbouring numbers in each dimension have one and only one bit different; (unit distance

characteristic)

(2) the first number and the last one in each dimension have one and only one bit different.(cyclic characteristic)

By Definition 7, we construct the two-dimensional code based on Johnson Code in Torus, which is encoded nodes of Torus. The method for constructing the two-dimensional code based on Johnson Code is described as the following. The binary data of one-dimensional Johnson Code which are transformed from 2k natural binary numbers of k bits in width ranged in natural binary number order serves as X-coordinate; the binary numbers transformed similarly from 2(n-k) natural binary numbers of remnant n-k bits in width serves as Y-coordinate. Where the X-coordinate of a node is combined with the Y-coordinate of it, the coordinate of any node is denoted the form as follows: N[n-1,…, k, k-1,…, 2, 1, 0], where " n-1,…, k " is Y-coordinate while "k-1,…,2,1,0" is X-coordinate. For example, to node with 5 bits width, the right 3 bits of it servers as the X-coordinate of it, and the left 2 bits Y -coordinate. It is shown as Figure 2.

The two-dimensional plane code based on Gray Code is homoplastically defined according to definition 7.

By the definitions above, we construct the two-dimensional plane code based on Gray Code in Torus, which is encoded nodes of Torus. The binary data of one-dimensional Gray Code which are transformed from 2^k natural binary numbers of k bits in width ranged in natural binary number order serves as X-coordinate; the binary numbers transformed similarly from 2^{n-k} natural binary numbers of remnant n-k bits in width serves as Y-coordinate. Where the X-coordinate of a node is combined with its Y- coordinate, the coordinate of each node is denoted the form as follows: N[n-1,…, k, k-1,…, 2, 1, 0], where " n-1,…, k " is Y-coordinate while " k-1,…,2,1,0 " is X-coordinate. It is shown as Figure 3.

Both the two-dimensional code based on Johnson Code and the two-dimensional code based on Gray Code in Torus are denoted the form as $A_{m+n-1\ldots}A_nA_{n-1}..A_1A_0$, where "$A_{m+n-1\ldots}A_n$" is Y-coordinate, while "$A_{n-1}..A_1A_0$ " is X-coordinate, respectively. The network topology formed by the two-dimensional code based on Gray Code is Torus network of $(2^n) \times (2^m)$ in size, any node of which has only four neighboring nodes. If Torus based on Gray Code increases 1 bit, the network increases $(2^n) \times (2^m)$ nodes. The network topology formed by the two-dimensional code based on Johnson Code is Torus network of (2n) × (2m) in size, any node of which has only four neighboring nodes. If Torus based on Johnson Code increases 1 bit, the network increases 4n or 4m nodes. Therefore, the two-dimensional code based on Johnson Code is provided with better utilizing rate of space and its scalable characteristic is more flexible than Gray Code.

At same time, in the Torus formed by the two-dimensional code based on Johnson Code (for $n \geqslant 2$, $m \geqslant 2$) , each node has only four direct links, which makes link number of whole network greatly decrease and cost of the network less than that of Hypercube network based on Gray code.

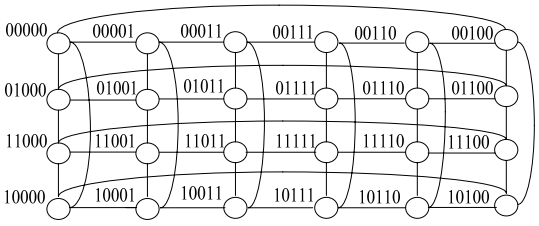

Figure 2. 4×6 two-dimensional code based on Johnson Code(n=5,k=3)

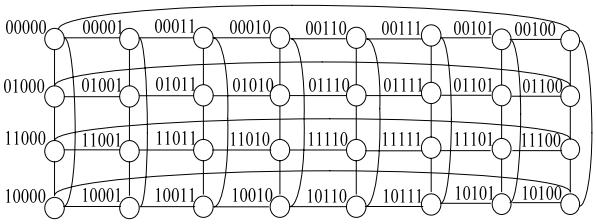

Figure 3. 4×8 two-dimensional code based on Gray Code(n=5,k=3)

V. COMPRESSING OF THE TWO-DIMENSIONAL NETWORK BASED ON JOHNSON CODE

The node code of the two-dimensional network based on Johnson Code can well adapt to Torus topology, but it requires more bits than the node address of natural binary code. With the increasing of the number of nodes, it consumes a great deal of storage resource, increases traffic and reduces the utilization rate of channel. For example, for 128×128 Torus structure, the code address of every node requires 128=128/2+128/2 bits and whole NoC requires the storage resource with 128×128×128=2M bits capacity. So it is necessary that the code be compressed. Two methods are put forward as follows.

A. Maximum Compressed Code

It makes compressed code bit number equal to that of the node address of natural binary code. Define function Compress_code=Transfer(Johnson_code), where function Transfer is the same as the description of Algorithm 2. The natural binary code of a node is Node_bin_code:

Node_bin_code=Transfer(Vertical_Johnson_code) ×2L+Transfer(Horizontal_Johnson_code),

where Vertical_Johnson_code is Y-coordinate part of node Johnson code defined in Definition 3 while Horizontal_Johson_code X-coordinate part, and 2L denotes the node number of every row. For example, in Figure 2, if the node code is 11011, X-coordinate code is 011 , Y-coordinate 11 and 2L=6. So, the node address of natural binary code is (1110)2=2×6+2=14. The compressed code bit number is equal to that of the node address of natural binary code.

B. Simple Compressed Code

Simple compressed code can simplify compressing logic and make compressed code bit number decrease to acceptable extent. In terms of Definition 3, any Johnson code ($T_{m-1}\ldots T_k\ F_{k-1}\ldots F_0$ or $F_{m-1}\ldots F_k\ T_{k-1}\ldots T_0$) is compressed the forms as follows: $a_1a_2a_3$, where a_1 with 1 bit width denotes whether the left part of Johnson code is

"T" or "F". If it is "T", a_1 is 0 or 1; a2 is the number of left "T" or "F" and the width of a_2 is $\lceil \log_2 N \rceil$ bits, and a_3 is the number of right "T" or "F" and the width of a_3 is $\lceil \log_2 N \rceil$ bits, where N is the total number of coded nodes. When the compressing method is applied two-dimensional code, "$a_1a_2a_3a_4a_5a_6$" expresses the compressed code of a node, where "$a_1a_2a_3$" denotes compressed X-coordinate code while "$a_4a_5a_6$" Y-coordinate.

When Torus with 2K×2L nodes is encoded by the two-dimensional network based on Johnson, the code length is K+L bits. The first compressed code length of it is $\lceil \log_2 (2K \times 2L) \rceil$ bits while the second that is $2+2(\lceil \log_2 K \rceil + \lceil \log_2 L \rceil)$ bits. The both of methods decrease code bit number from O(N) to $O(\log_2 N)$. But in implementation logic, the second method is simpler than the first one, and easy hardware implementation.

VI. NoC ROUTING ALGORITHM

Utilizing the two-dimensional code based Johnson Code, the nodes of NoC is encoded, its process implies all the information of building routing. The node code is used as label and the distance between the source node ($N=N_{n+m-1}\ldots N_k\ldots N_m\ N_{m-1}\ldots N_i\ldots N_0$) and the destination node D is d=Haming(N ⊕ D). The addresses of the four circumjacent nodes of a node are obtained by both the address of the node and the characteristic of the two-dimensional code based Johnson Code and is described as the following:

left_node_id=$N_{n+m-1}\ldots N_k\ldots N_m\ \overline{N}_0\ N_{m-1}\ldots N_i\ldots N_1$;

right_node_id=$N_{n+m-1}\ldots N_k\ldots N_m N_{m-2}\ldots N_i\ldots N_0\ \overline{N}_{m-1}$;

up_node_id= $\overline{N}_m\ N_{n+m-1}\ldots N_k\ldots N_{m+1}N_{m-1}\ldots N_i\ldots N_0$;

down_node_id=$N_{n+m-2\ldots}N_k\ldots Nm\ \overline{N}_{n+m-1}N_{m-1}\ldots N_i\ldots N_0$.

The definition above shows the addresses of the four circumjacent nodes of any node are generated by simply implementation of register shift. The addresses of circumjacent nodes of a node based on (x, y) coordinate are related to origin of coordinates. At same time, the judgment is made that whether the node is on the edge defined by person. (in natural, Torus network has no the node on the edge). The node address of Torus network of the two-dimensional Johnson Code can be dynamically changed. The node address shows relative position between nodes. By contraries, the node address based on (x, y) coordinate denotes absolute position. Coordinates transform result in routing mechanism transformation.

There are the X-Y routing, the interlaced X-Y routing and the DyAD routing in Torus. X-Y routing is simple in implementation and widely used. By utilizing both the two-dimensional code based Johnson Code and the deterministic peculiarity of X-Y routing, an improved algorithm for X-Y routing is proposed and can decrease the compute complexity of routing and hardware cost. In the algorithm, The two bits which are called bit_x and bit_y used as direction identifiers, are added in packets. If bit_x equals 0 or 1, packets move to the left or the right. Similarly, if bit_y equals 0 or 1, packets move to the up or the down. Let the direction of output ports is expressed out_dir.

Algorithm 3 The improved algorithm for X-Y routing is described as follows by pseudocode.

```
begin
// set direction flag and deliver packets from source node
  if（the given node is source node）
    { // set direction flag
      if （ Haming （ left_node_id ⊕ D ） <= Hamin
        （right_node_id ⊕ D））
          bit_x=1;
      else
          bit_x=0;
      if （ Haming （ up_node_id ⊕ D ） <= Haming
        （down_node_id ⊕ D））
          bit_y=1;
      else
          bit_y=0;
     // deliver packets from source node
      if (N_{m-1}…N_i…N_0 ⊕ D_{m-1}…D_i…D_0!=0)
          if(bit_x==1)
              out_dir= left
          else
              out_dir= right;
      else
          if(N_{m+n-1}…N_k…N_m ⊕ D_{m+n-1}…D_k…D_m!=0)
              if(bit_y==1)
                  out_dir=up;
              else
                  out_dir=down;
          else out_dir=home;
  deliver packets from source node in out_dir;
  }
  //deliver packets from middle node
  if（the given node is middle or destination node）
  {  if (N_{m-1}…N_i…N_0 ⊕ D_{m-1}…D_i…D_0!=0)
          if(bit_x==1)
              out_dir= left
          else
              out_dir= right;
     else
          if(N_{m+n-1}…N_k…N_m ⊕ D_{m+n-1}…D_k…D_m!=0)
              if(bit_y==1)
                  out_dir=up;
              else
                  out_dir=down;
          else
              out_dir=home;
    deliver packets from the middle node in out_dir;
   }
  end
```

When nodes are labeled with coordinate (x, y) in X-Y routing, routing on nodes is implemented by arithmetic operations. The improved algorithm for X-Y routing on middle nodes requires only three or six logic operations, where a logic operation find out whether bit_x (bit_y) is 0, and that on source node does more four nonequivalence operation and two compare operation than middle nodes.

Logic operation and shift operation are simpler and require less time in implementation than arithmetic operation. At same time, Logic operation and shift operation require small area cost of VLSI. Similarly, the traversing based on Y-X routing, the interlaced Y-X routing and the DyAD routing are also implemented.

VII. THE DESIGN FOR NODE STRUCTURE

The NoC adopts Torus topology. The node structure of NoC which is designed is shown as Figure 4.
The node includes SWITCH, four queues (FIFO) which receive respectively asynchronous input in four directions, IP and CLK_GEN which generates clock. The data which is received or sent by nodes is one with many bits that include the address utilizing the two-dimensional code based Johnson Code. The wires which send data are labeled tr and those that received data rec. Synchronous signal wires have sync_tr and sync_rec which denote respectively send signal and receive signal. The rec_full and tr_empty express respectively the signal whether input queue is full and one whether output queue is empty. FIFO writes according as the clock of neighboring node while it reads according as the native clock.

Every node has an IP which switches data with SWITCH module utilizing the native clock. In the NoC, every node links four neighboring nodes and there are bidirectional data links between two linked nodes. Each node has a clock and there is asynchronous communication between two nodes (GALS). Routing makes use of the improved algorithm for Y-X routing above. In nodes, centralized control is adopted and one module controls data sending and receiving. According as impartial strategy, packets in four FIFO are switched in turn.

VIII. PERFORMANCE EVALUATION

One key aspect about NoC is the performance evaluation. To compare and contrast different NoC architectures and algorithms, a standard set of performance metrics can be used. For example, it is desirable that an MPSoC interconnect architecture exhibits high throughput, low latency, energy efficiency, and low area overhead [18]. Generally, the performance metrics of the average network latency and the average network throughput are of great importance [19]. To evaluate the improved algorithm for Y-X routing of 2-d Torus utilizing two-dimensional plane code based on Johnson Code. we compared it with the algorithm for Y-X routing of 2-d Mesh by simulating in three different network sizes(16,32,64 nodes).

We have developed a discrete event, cycle accurate NoC simulator. It provides substantial support to experiment with NoC design in terms of routing algorithms and applications on various topologies. The simulator is written in SystemC. The wormhole packet switching is adopted in two topologies, Torus and Mesh

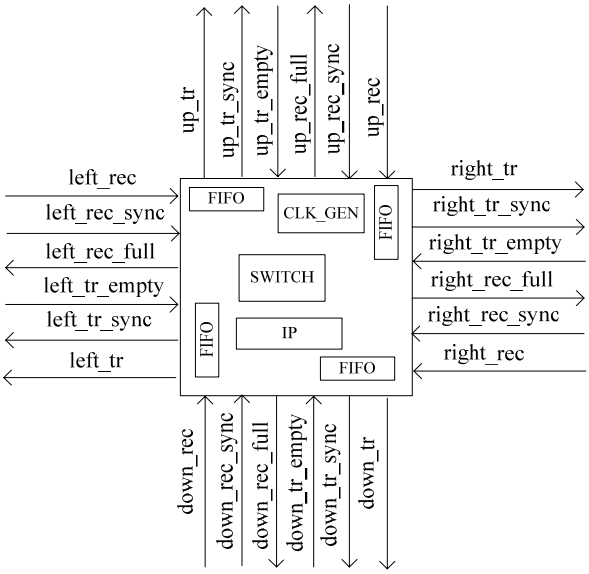

Figure 4 Node structure

nodes have been defined with the same node architecture showed by figure 4 , which ensure the justness of analyses.

Each node has an external network interface to connect the IP core to the NoC. The external IP core can act as a packet source and/or as a packet destination (sink) depending on the simulated scenario. In our simulations, each source IP core generates packets and sends them to other IP cores. Each packet has three 32-bit flits (flow control unit, flit). The first (head) flit of a packet is sent to the routing mechanism of the node, and then transferred on the output of the target channel (if the input channel of next node is room). Once the head flit has been processed by the routing element of a node, a switching mechanism is defined to forward all immediately following packet-flits to the outgoing links of the target path to the destination node. We changed the flit rate injection from 0.05 flit/cycle/node to 0.5 flit/cycle/node. Each input channel of each node consists of one virtual channel. Each virtual channel of the input channel consists of 8 flits FIFO buffer. Each output channel consists of one flit buffer. The clock frequency of NoC is 1 GHz.

The modes simulating have single hot-spot destination scenario, double hot-spot destinations scenario, homogeneous destinations scenario and so on. we simulated and analyzed the performance by homogeneous destinations scenario which can show the main performance of NoC.

Figure 5 shows the average latency obtained by Torus, Mesh topologies under homogeneous source and destination distribution scenarios. All the nodes behave like sources and can be addressed as destination for packets with uniform probability distribution. Latency is shown as a function of the number of nodes N and the injection rate parameter of multiple source nodes. The result shows that the average latency of Torus is smaller than Mesh.

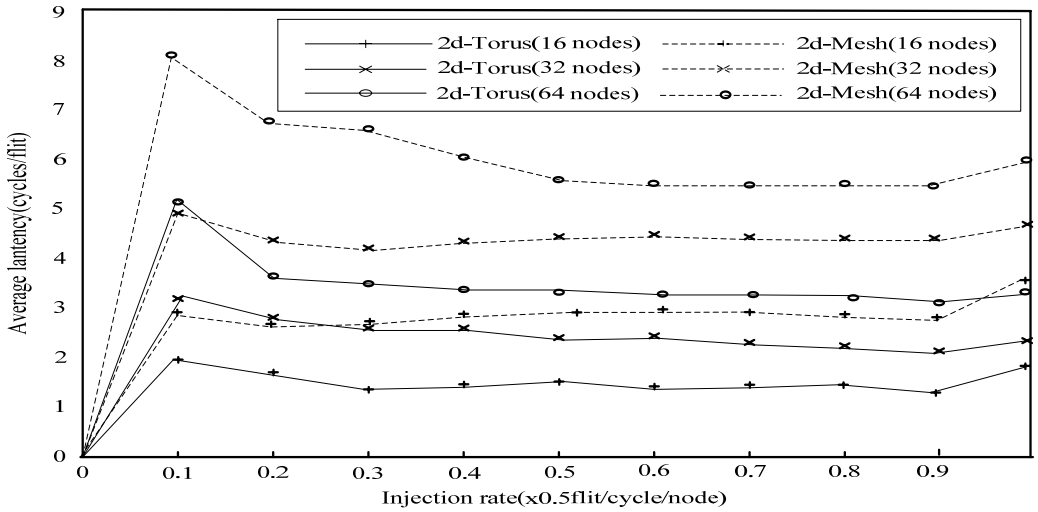

Figure 5. Average latency

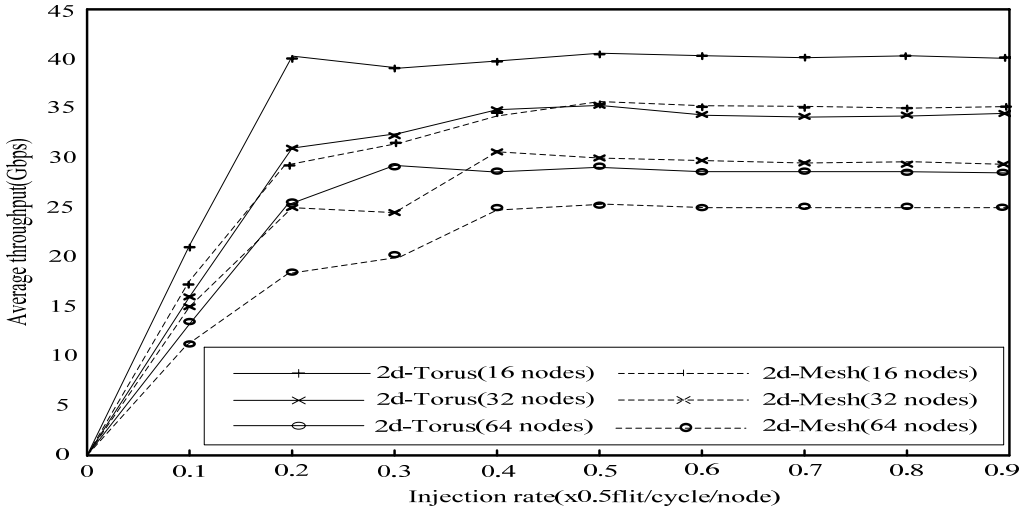

Figure 6. Average throughput

Figure 6 shows the throughput results with respect to the NoC topology, the relevant routing algorithms and the number of nodes, under homogeneous scenarios with uniform distribution of sources and destinations. When all node sources increase the injection rate, the average latency and the average throughput increase with the injection rate of source nodes, up to the saturation of network. This performance parameter shows Torus outperform Mesh.

IX. Conclusions

The paper analyzes the topology and the node code of NoC, by the combination of NoC topology with corresponding node encoding , and proposes a two-dimensional code based on Johnson Code which implies the relation between neighbouring nodes and their links and the global information of routing. At same time, the two methods for code compressing are presented to reduce the storage space of node address and increase. The improved algorithm for X-Y routing is presented which is implemented with three or six logic operations in middle nodes and The node structure of NoC is designed. The paper compares the improved algorithm for Y-X routing of 2-d Torus utilizing two-dimensional plane code based on Johnson Code with the algorithm for Y-X routing of 2-d Mesh by simulating in three different network sizes, and obtains that performance parameter of the improved algorithm for Y-X routing of Torus outperform that of Mesh. The code proposed in the paper is useful not only in NoC routing but also in NoC task mapping.

ACKNOWLEDGMENT

It is a project supported by the National Science Foundation of China under grant No.90607008 and Xi'an Applied Material Innovation Fund Project (XA-AM-200615).

REFERENCES

[1] Tobias Bjerregaard, Shankar Mahadevan, “A Survey of Research and Practice of Network-on-Chip,” ACM, 2006.

[2] L. Benini and G. De Micheli, "Networks on chip: a new paradigm for systems on chip design," in Proc. Design Automation and Test in Europe (DATE), 2002, pp. 418-419.
[3] M. Horowitz and W. Dally, "How scaling will change processor architecture," Digest of Technical Papers. ISSCC. 2004 IEEE International Solid-State Circuits Conference, vol. 1, 2004, pp. 132-133.
[4] L. Benini and G. De Micheli, "Powering networks on chips: energy-efficient and reliable interconnect design for Socs," in Proc. the 14th Int. Symp. on System Synthesis, pp. 33-38, 2001.
[5] F. Moraes, N. Calazans, A. Mello, et al, "HERMES: an Infrastructure for Low Area Overhead Packet-Switching Networks on Chip", Integration, the VLSI Journal, vol. 38-1, 2004, pp. 69-93.
[6] G. Ascia et. al, "Multi-objective mapping for mesh-based NoC architectures", In Proc. CODES, 2004, pp. 182- 187.
[7] J. Hu, R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," IEEE Trans. on CAD of Integrated Circuits and Systems, 24(4), pp. 551-562, April 2005.
[8] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in Proc. the Design Automation Conference, Las Vegas, NV, pp. 684-689, 2001.
[9] Ahmed Hemani, Axel Jantsch, Shashi Kumar, et al, "Network on chip: An architecture for billion transistor era," In Proceeding of the IEEE NorChip Conference, November 2000, pp. 166-173.
[10] Ville Rantala., "Network on Chip Routing Algorithms", TUCS Technical Report. No 779, 10-12. 2006.
[11] Peter K. Loh, Wen Jing Hsu, Yi Pan, "The Exchanged Hypercube", IEEE TRANS. Parallel and Distributed Systems, VOL. 16, NO. 9, 2005.
[12] Ananth Grama, "Introduction to Parallel Computing" (Second Edition). Pearson, 2003. pp 55-5.
[13] Amawy,.& Latifi, "Properties and performance of folded hypercubes," IEEE Trans. on Parallel and Distributed Systems 2, 31-42, 1999.
[14] Erno Salminen, Ari Kulmala, and Timo D. Hamalainen. "On network-on-chip comparison," 10th Euromicro Conference on Volume, Issue, 29-31 Aug. 2007, pp. 503-510.
[15] Jingcao Hu, "DyAD-Smart Routing for Networks–on-chip," in Proc.41'st ACM/IEEE Design Automation Conference, San Diego, CA, 2004.
[16] M.Dehyadgari, "Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC," The 17th International Conference on Microelectronics, 13–15 December 2005:204-208.
[17] M.Heiss, "Error-Detecting Unit-Distance Code," IEEE Trans. Instrumentation and Measurement, Vol 39, No 5. October 1990.
[18] Partha Pratim Pande, Grecu C, Jones M, etc. "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," Transactions on Computers,Volume 54, Issue 8, Aug. 2005, pp. 1025 – 1040.
[19] Luciano Bononi, Nicola Concer. "Simulation and Analysis of Network on Chip Architectures: Ring, Spidergon and 2D Mesh," in proc. of the design, automation and test in Europe, 2006, pp. 154-159

Yang Xiaoqiang was born in 1971. He received B.Eng. degree in computer science and technology from Liaoning Technical University, Liaoning China and M.Eng. degree in computer and technology from Xidian University, Xi'an China in 1995 and 2003 respectively. He is currently pursuing PhD degree in Micro-Electronics from Xidian University, Xi'an China .

He works as a lecturer at Xi'an University of Science and Technology. His current research interests include parallel algorithms, parallel architecture, SoC Design and Verification.

Li Junmin was born in 1956. He received the B.S. degree in computer application from Shandong University of Science and Technology in 1982. He is an associate professor and Director of computer science and technology.

He works as a lecturer at Xi'an University of Science and Technology. His current research interests include parallel algorithms, parallel architecture, computer architecture and SoC.

DU Huimin was born in 1966. She received the M.S. and PhD degree in computer science the Northwestern Polytechnical University, Xi'an, China, in 1992 and 2000 respectively. From 2000 to 2002, she got a postdoctoral position in Northwestern Polytechnical University.

She is currently Professor, Director of Micro-Electronics Research and Teaching Group at computer science department at Xi'an Institute of Posts and Telecommunications, Xi'an , China. Her research interests are mainly on ASIC design, formal verification for hardware design, etc.

Han Jungang was born in 1943. He received the B. Eng. in Mathematics Department of Jilin University, China, in 1966 and M.S. in Institute of Computing Technology, Chinese Academy of Science, Beijing, P.R. China, in 1981.

From 1982 to 1993, he is an associate professor, Xidian University, Xi'an China. He is a visiting scholar, Computer Science Department, University of Calgary, Canada, Oct. 1986 to Sept. 1988. Since 1994 he has been on the faculty of the Department of Computer Science at Xi'an Institute of Post and Telecommunication, where he is a professor, director, doctor tutor, and Head of Department of Computer Science, Xi'an Institute of Post and Telecommunication.

He is active in formal design and verification of SoC, application technology of computer etc. He has published numerous journal and conference articles. From 1999, Professor Han received Chinese Government special subsidy,. He is a senior member of Chinese Computer Federation, Chinese Communication Federation, special committee of CAD&CG, CCF.