Abstract—In Network on Chip effectiveness of router is dependent on the buffer locality; which makes the efficient flow control. Previous buffer design of Triplet-Based Hierarchical interconnection network (TBHIN) is standard, which leads to insufficient accessibility of this decisive resource, where each virtual channel owns a fixed number of buffers. In this article the design is implemented with sharing the buffers among the virtual channels, to improve the performance. The cyclic queue is allowed the simultaneous access to the shared buffer, which is one of the characteristics of TBHIN. A cycle-accurate simulator is used to obtain packet latency and throughput results for conventional and shared buffer designs. Simulation results illustrate that the packet latency is reduces up to 29% by shared buffer design in comparison to conventional buffer design. Also shared buffer design improves throughput up to 11.87% over the conventional buffer design.

Keywords— TBHIN, Shared buffer, performance evaluation, Network on chip

I. INTRODUCTION

Packet-switched network-on-Chip (NoC) becomes the interconnection fabric in both the general-purpose multi-processor chips and the application specific systems-on-chip. So for traversing packets smoothly and avoid traffic congestion, efficient buffer design is obligatory.

Buffers provide spaces for arriving flits, so an efficient buffer designing is considered an important factor in the flow control design of router architecture. In buffer designing lot of research is done such as some have focused on the input buffer design [1], while the others have intention toward output buffer design [2]. Some researchers are concerned about energy-saved [3-5], high-throughput [6-8], self-compacting[9], efficiency [10]and shared [11-14] buffer design. Generally the architects pay attention to general structure of on-chip design [15] whereas several strived to design shared buffer for real-time system [16]. In NoC mostly the buffers are design keeping the 2D-Mesh topology in mind, however our architecture topology TBHIN is different than the 2D-mesh topology. So the buffers are designed considering the characteristics of TBHIN.

Buffer is an important unit in the design of router architecture. Shared buffer is widely adopted in the design of router. In [8, 11-14] have discussed the shared buffer design in detail.

Soteriou et al.[2] discuss the high-throughput shared buffer design. They show the advantages and disadvantages of input-buffer router and output-buffer router. They proposed a new router design that aims to emulate an output buffer design including efficient pipeline and novel flow control. Our design is more concerned about the input shared buffer design.

Wei et al.[11] present HiBB shared buffer in NoC router based on Hierarchical Bit-line Buffer. HiBB has low power inherent attribute and flexible configuration according to the traffic. Zhang et al.[12] propose network on chip DAMQ-FP dynamically shared buffer to decrease the memory and area requirement of statically allocating shared buffer among multiple virtual channels. It can decrease the latency and get high throughput. Zhang et al. [13] discuss the shared buffer design and evaluates it using an application of JPEG decoding. Their observations provide evidence that memory resources can be saved distinctly using shared buffer method. Tran et al. [14] are motivated from the scenario of regular workload in which huge numbers of buffers are idle whereas some are always busy. In their router design buffers are shared with multiple input ports to improve the performance. Different from their design, our design of shared buffer is oriented to TriBA where we adopt a new controller to control the buffer allocation.

This article illustrates the shared buffer design in TBHIN on each port, compared to fixed number of buffers for each virtual channel. The detailed design of shared buffer for TBHIN is presented in this paper and employs an address queue to control the allocation and de-allocation of buffer spaces. The buffer structure is arranged into linked list structure in which the pointer at head and the pointer at tail are used to control the read and write.
The remaining part of article is prepared with following sections. Background and motivation is described in section 2 along with TBHIN and problem statement. Section 3 describes proposed mechanism of sharing buffers for TBHIN. Results are illustrated in section 4 and the article is concluded in section 5.

II. BACKGROUND AND MOTIVATION

TBHIN is a new hierarchical and scalable topology in Network on Chip, which has many advantages over other NoCs, such as simple topology, full-connectivity for locality, and the scalability to build a large-scale TBHIN [17, 18]. Three levels of TBHIN are shown in Figure 1. Lower level to Upper level TBHIN is constructed, based on fundamental module, recursively with replacement of nodes in basic structure. It could be observed from Figure 1(c) which is extended from 2nd level shown in Fig 1(b). A distributed deterministic routing algorithm (DDRA) [19] is used for routing in TBHIN.

Based on TBHIN topology, Triplet-Based Architecture (TriBA) was proposed as an object oriented architecture[17], which has communication locality feature [20]. Messages are passing between cores. TriBA consists of three parts: InterUnit, ProcUnit and DataUnit, where InterUnit is responsible for the interconnection between cores. The work of this paper is associated with the InterUnit. Shared buffer is one of the most important parts to implement the performance of InterUnit.

III. SHARED BUFFER DESIGN

Buffers have the efficient functionality in basic infrastructure of router architecture design. They are common to provide spaces for arriving flits. Some researchers express that in NoC router buffers are area occupants and the largest leakage power consumers, 64% is observed consuming power leakage of total router in[21] and in [22] states that buffer space is dominated.

TriBA, as an object-oriented architecture, has high requirements for message storage spaces. It needs to store as many messages as possible. In order to achieve this goal, shared buffers are designed in each direction. All the virtual channels in each direction own a shared buffer with a buffer depth of 8. For TriBA, there are four directions including the local port. This implies that there are 32 buffers plots for all four directions. The advantage of such design allocates buffers to each arriving flit according to the need of each direction. If a virtual channel needs to store more messages, then it is allocated more buffer spaces. If one virtual channel

(a) 3-node(K=1)
(b) 9-node(K=2)
(c) 27-node(K=3)

Figure 1 The number of nodes in a K TBHIN is 3^K, where K is the level of TBHIN and n=1,2,3,...

A. Problems Statement

Despite advantages of TBHIN, it has critical pathway in the network. For a 9-node TBHIN, if any node A/B/C of one triplet needs to send packets using its default DDRA routing algorithm to another triplet’s node G/H/I, those packets have to traverse on the link between node C and H. It is obvious this is the only link for traversing the packets during the communication, which will lead to increase of the injection rate and causes congestion problem due to more interchange. Links between node B to D and node F to G also suffer in this situation. Moreover, link between nodes B to C will have the possibility of blockage because node B uses the same link to node C/H/G/I. Also the same link would be occupied too, if the node C receive packets from the node D/E/F.

Figure 2 The critical pathway

Links between node D to F and node H to G also face the same situation. So there exists a round critical pathway, which might cause the serious collisions in the network traffic. Figure. 2 illustrates the critical pathway.

The fixed number of buffers cannot exert the TBHIN performance, because the path in the critical pathway will consume more buffers than the common path. It will lead the node to be the critical pathway and become the crowded node during the communication. So we adopt the shared buffer design for TBHIN to make the buffer availability more efficient.
needs to store fewer messages, then it is allocated less buffer spaces. We select 8 buffers for each direction because more buffers will occupy large area of router design and fewer buffers will lead to low efficiency. The number of buffers is selected according to the characteristic of TriBA.

Buffer could be arranged at the front or back of crossbar, and is named input buffers or output buffers respectively. Output buffers are attractive because they have lower latency and higher throughput under high workloads. However, output buffers suffer from the requirement of high clock speed to match the four port input and the power budgets. These prohibitive factors of output buffers have led to adoption of input buffer and therefore selected for our design.

To implement the shared buffer mechanism, the detailed buffer structure and its controller are shown in Figure 3. This structure includes receiving module, four buffer queue named Queue_i (i=0,1,2,3), one shared buffer controller named AddrQueue, and the sending module. The receiving and sending module are charged for receiving data and sending data. The Queue_i includes the queue Queue_0 for the local port, and Queue_1 for the top port, Queue_2 for the left port and Queue_3 for the right port. Each queue Queue_i has the read and write operation units, which are responsible for reading data from and writing data into buffers. Note that only the addresses of the data are written to the buffer. The address item is the actual address that will be stored in Queue_i. The crossbar is similar to the conventional design so that part doesn’t include in the Figure.

AddrQueue is a queue controller with 5-bit address length, that is, it has 32-flit buffer depth. The task of AddrQueue is to allocate and set the buffers free to use. Figure 4 shows the detailed design of AddrQueue. The valid write buffer position is that, which would be unoccupied items in AddrQueue. The AddrQueue consists of data item and address item. The data item is the messages stored in buffer. The address item is the actual address that will be stored in Queue_i. The way of AddrQueue arrangement prefers the indirect address. Each Queue_i is arranged like a linked list which has a pointer on tail and on head. When the network receives a message, it will be stored into the queue AddrQueue, and put the corresponding address to which the head pointer is pointing to.

If the message is stored successfully, the head pointer will increase its position counter. When sending out a message, the message address will be gotten, which will be pointed by the tail pointer and the data will be deleted from the buffer. After the message is sent out successfully, the tail pointer will increase its position counter accordingly.

Note that AddrQueue is a controller with set of 4 input and 4 output channels. All these channels read and write these units in parallel without interleaving for each other. It requires the AddrQueue running with a higher clock frequency. All the shared buffers are arranged into a queue. The read and write operation actually write the address of data to the buffer which makes each data in queue Queue_i of the same length.

As mentioned earlier, TriBA is an object-oriented architecture and its network transmits well-structured object message. Object message is of fixed length with 32-bit. Each unit is arranged into same length thus an object message can only be arranged to a unit. We use XXH to represent the available unit and waiting for a massage to be inserted into it.
IV. SIMULATION

Noxim [23], a cycle-accurate simulator developed by System C is used for conducting the simulation. The available version of Noxim is oriented to 2D-Mesh. The TBHIN topology is different from 2D-Mesh topology; so it is modify to build the TBHIN interconnection network. The main modifications include: 1) each node degree. Except the local port each node in the TBHIN has degree 3. Each port is labeled as 0, 1, and 2 whereas 3 is labeled to the local port. 2) Ports connection. To route TBHIN, we adopt the DDRA routing algorithm to run the Noxim simulator. 3) Signal invalid. We disable the links of three vertexes and make the corresponding signals Req and Ack invalid. A 9-node TBHIN is built through the above modification.

The related parameters of Noxim are configured as shown in Table 1. Four traffic patterns are selected to evaluate the design including transpose, random, shuffle and bitreversal.

High throughput and low latency parameters have the imperative effect in the design of NoCs and buffers play an important role to achieve these goals. In this section, the latency and throughput simulation results are illustrated.

A. Average traffic latency

Figure 5 presents the average latency under different injection rates. All the data are 3 identical runs. Clearly, the proposed shared buffer design outperforms the non-shared buffer design (here it is referred as conventional design) under all traffic patterns. Packets are injected from each node to arbitrary scattered destinations into the network assuming as random traffic. As compared to the conventional design, the shared buffer design reduces up to 28.99% average latency in random traffic pattern. The message distribution is representing with a traffic matrix for the transpose traffic pattern. A 3x3 matrix is constructed with three rows, for 9-node TBHIN. The average latency reduces up to 29% in transpose traffic pattern. The shared buffer design reduces average latency up to 26% in the bitreversal and 12.07% in the shuffle traffic patterns compared to the conventional design. Overall, the shared buffer design outperforms the conventional design for all four traffic patterns.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic mode</td>
<td>Random, Transpose, Bitreversal, Shuffle</td>
</tr>
<tr>
<td>Buffer depth</td>
<td>Shuffle</td>
</tr>
<tr>
<td>Simulation cycles</td>
<td>8 flits</td>
</tr>
<tr>
<td>Warmup cycles</td>
<td>10000 cycles</td>
</tr>
<tr>
<td>Flit size</td>
<td>1000 cycles</td>
</tr>
<tr>
<td>Number of virtual channel</td>
<td>8</td>
</tr>
<tr>
<td>Number of port</td>
<td>4</td>
</tr>
</tbody>
</table>

B. Throughput

The number of flits accepted per cycle is measured as the throughput. High throughput commonly occurs when the channels become saturated in the network. Figure 6 presents the throughput results. The throughput increases consequently as the injection rate increased. The throughput will not be rising any more when the injected rate arrives at peak value. The throughput graph is represented in the same trend as in the average latency testing. Overall, our proposed shared buffer design can outperform the non-shared buffer design by 8.34% for transpose, 12% for random, 5.19% for shuffle and 8.2% for bitreversal traffic pattern under the saturation throughput.

For latency and throughput experiments, shared buffer design outperforms non-shared buffer design. Simulation results show that shared-buffer is better than non-shared buffer design for TBHIN.
V. CONCLUSIONS

To handle increased number of messages TriBA architecture requires higher number of buffers. This paper proposes the shared buffer design for TriBA interconnection network TBHIN and to make the availability of buffer more efficiently. The detailed design is illustrated in this paper. Simulation results using a cycle-accurate simulator shows that shared buffer design reduces up to 29% packet latency and improve up to 11.87% throughput over the non-shared buffer design. The result shows that shared buffer design is more suitable to TBHIN.

ACKNOWLEDGMENT

This work is partially supported by National Nature Science Foundation of China.

REFERENCES