Analysis of Ring Topology for NoC Architecture

Avinash Kamath  
Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal  
kamath93@gmail.com

Gaurangi Saxena  
Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal  
saxenagaurangi94@gmail.com

Basavaraj Talawar  
Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal  
basavaraj@nitk.edu.in

Abstract—In recent years, Network on Chips (NoCs) have provided an efficient solution for interconnecting various heterogeneous intellectual properties (IPs) on a System on Chip (SoC) in an efficient, flexible and scalable manner. Virtual channels in the buffers associated with the core helps in introducing the parallelism between the packets as well as in improving the performance of the network. However, allocating a uniform size of the buffer to these channels is not always suitable. The network efficiency can be improved by allocating the buffer variably based on the traffic patterns and the node requirements. In this paper, we use ring topology as an underlying architecture for the NoC. The percentage of packet drops has been used as a parameter for comparing the performance of different architectures. Through the results of the simulations carried out in SystemC, we illustrate the impact of including virtual channels and variable buffers on the network performance. As per our results, we observed that varied buffer allocation led to a better performance and fairness in the network as compared to that of the uniform allocation.

Key words— NoC; buffers; virtual channels; ring topology

I. INTRODUCTION

The growing desirability for low-power and high performance of the computation intensive applications has led to an increase in the number of computing resources on a single chip. Therefore, ICs integrating several heterogeneous resources on a single chip, commonly known as system on chips (SoC) are becoming increasingly popular. However such integration on a single chip can often become complex and can hence, make the interconnection between different resources and IP a challenging task [1]. Several approaches have been followed to overcome the complexity of interconnecting different heterogeneous resources. NoCs is one such technological approach which aims at improving the scalability and providing high performance to the SoC networks. NoCs are preferred over other interconnecting methods like dedicated wires and buses due to its better reusability, flexibility and scalability of bandwidth.

Dedicated wires are helpful for the systems having a small number of cores. However, as the system complexity increases, the number of wires around each core increases. The use of dedicated wires can hence lead to poor flexibly and make the physical system cumbersome. The use of buses can overcome the inflexibility of the dedicated wires. But the use of buses can result in lower throughput since it allows only one communication transaction at a time. This in turn results in increased packet latency. Use of multiple interconnected buses can overcome some of these problems but the scalability provided by the buses is still limited [2].

Buffers and channels are the two major assets of the interconnection networks. Each channel is associated with a single buffer. This can lead to packet congestion in some channels and in turn bring down the overall throughput of the system. However, virtual channels can overcome this issue by providing a way
for multiplexing a single channel into multiple buffers. By using VCs, multiple packets can share a particular physical channel at the given point of time. The use of VCs enhances the resource allocation of packets and the overall throughput of the network, and reduced the network latency [3].

Although VCs help in reducing the network latency and improving the bandwidth, there is a need for a routing algorithm that helps in reducing the packet loss when some channels are being heavily used as compared to others. In real world scenarios, the number of packets in the virtual channels of a physical channel may significantly differ. A particular channel may have considerably lesser number of packets to deliver than the other channels. Hence, having the same buffer size for each virtual channel or following the normal packet allocation may not be desirable [4]. Allocating the appropriate buffer size for each channel can prove to be helpful in such scenarios. In such scenarios, uniform buffer allocation may lead to packet loss when the traffic in different channels is varied.

In this paper, we use ring topology for the interconnection of different IPs in an on chip network. We demonstrated the importance of using virtual channels over single channels by comparing the performance of both the implementations. We tried to modify the design of traditional ring topologies by making changes like including bidirectional links and variable buffer in order to improve the performance of the network using ring topology. Further, we illustrate the importance of having a fair way of allocating buffers based on the traffic pattern. The performance of different designs is compared by taking their respective packet losses into account.

Crossbar topologies are more widely used than any other topology as an underlying architecture for NoCs. Crossbars provide low packet latency and a fair way of delivering the packets. However, the amount of physical resources used in crossbar is considerably higher than that used in the ring topologies [5]. Our next aim and ongoing work is based on improving the performance of the ring topologies by making its performance comparable to that of the crossbar technologies.

The rest of the paper is organized as follows: section II discusses the related work done in this area, section III introduces the design and methodology followed in the paper, experiments and results have been discussed in section IV and lastly the paper is concluded in section V.

II. LITERATURE SURVEY

In [6], authors have compared the performance of NoC with the traditional point to point and bus communication architectures. The area covered by different architectures and their respective energy consumption were also taken into account. Real world workloads like video applications etc. were considered for assessing the performance of these architectures. Through these experiments it was revealed that the NoC architecture scaled better than the other traditional architectures in terms of energy, area as well as the performance.

The authors of [7] discuss the necessity of having a programmable interconnection network for the computation intensive and complex applications. NoC architecture fulfills the demands of having a high interconnect bandwidth and for exploring the parallel processing capacity of multiple computational resources.

In [8], authors compare the performance of different networks with and without VCs. By conducting experiments on the 2D meshes of different sizes, authors reveal that the improvement in latency after including VCs is higher for large grids than that of the small dimension networks. Also, the packet injection rate increases significantly when VCs are included in an NoC architecture. Overall, the performance of NoC improves with the insertion of virtual channels. Similar results were obtained in [9], wherein the authors carried out simulations to bring into light the enhancement in performance after including VCs. From the results of the simulations it was drawn that virtual channels are ideal for the NoCs, especially for the ones with high packet injection rates. The decrease in the packet latencies after including VCs occurred with the equivalent increase in power consumption. It was concluded that the NoC with high packet injection rates should consist of more number of VCs as compared to the one with lower injection rates. In the latter case, VCs should be optimized for both leakage and dynamic power consumption.
In [9] the authors propose a VC allocation algorithm for efficiently assigning the VCs based on the traffic requirements in a 2D mesh. The NoC architecture following this algorithm performs better than the uniform allocation method in terms of buffer utilization.

A centralized buffer structure has been introduced in [10], which dynamically allocates the number of virtual channels and buffers based on the traffic conditions. The simulations were carried out on a conventional NoC architecture. Various traffic patterns like uniform random, tornado and normal random were used to evaluate the performance of the architecture proposed but the authors. Network latency and the buffer utilization were used to compare the performance of the designs with and without the centralized buffer. However, the throughput and the percentage of packet losses were not taken into account in this paper, for assessing the performance of the modification proposed by the authors.

In [12] authors discuss about the need of fair allocation of resources in a network. Different techniques are discussed to compare their efficiency. The min max approach fits our ‘algorithm’ because it does not allocate equal quantity of resources to all the switches but instead allocates them based on the traffic requirements. Jain’s fairness is discussed in [15] to measure fairness in a system. We have used this measure to evaluate fairness in different routing methods.

### III. DESIGN AND METHODOLOGY

The design chosen for the simulations consists of a register ring of size 5. This means register 1 is connected to register 2 through a link, register 2 connected to register 3 and so on till register 5. Register 5 will be connected to register 1, forming a ring of registers. Each register will have its own input and output buffers. Each buffer can accommodate a certain number of message packets at the given time. Each message packet includes information like source core, destination core and the message to be delivered. These buffers are connected to cores. Thus each core can be associated with a set of input and output buffers and also a register present in the register ring. So if a core wants to send a message to another core, it has to add it to the input buffer connected to it. This message gets transferred only if the buffer has a vacancy. If the input buffer is full the packet gets dropped. Later, the buffer passes the message to the ring register that it is connected to, only when the register is empty. The
register will pass the message to next register connected to it. This message keeps travelling from one register to another till the register assigned to the destination core gets the message. Once the message reaches this register, it transfers it to the output buffer and then the output buffer transfers the message to the core connected to it. We have considered the size of each buffer to be 3 i.e. each buffer can accommodate 3 message packets at a given time. The method mentioned involves the simplest design and routing. These things can be modified in order to achieve better performance.

One variation from the usual way of having input buffers is using the idea of virtual channels. This means each input buffer has many sections of same size. Each section will store the packets having a certain core as destination. Earlier, if most of the packets had a specific core as destination say d, they would occupy the input buffer of a given core most of the time leading to the dropping of packets having other cores as destination. Now, the packets meant for other cores will be stored in a separate section. So the packets having destination as d will have a different section in the input buffer and they will have to compete among themselves. In this design, the situation where packets with destination d bully the packets with other cores as destination can be avoided. But if packets having destination d are large in number then the number of dropped packets will still section in the input buffer and they will have to compete among themselves. In this design, the situation where packets with destination d bully the packets with other cores as destination can be avoided. But if packets having destination d are large in number then the number of dropped packets will still be high. So, this idea can only avoid the bullying but not the number of packet drops.

Another idea proposed involves the use of virtual channels but in a different way. Here the size of the sections assigned is not the same. This means the size of the section allotted for packets with destination a need not be the same as the size of the section meant for packets with destination b. This idea is used because if we assign same size to the sections meant for different destinations, there will be wastage of space. For example, say a very small fraction of packets have a as destination and a large fraction have d as destination. Now, the size of the section meant for packets with destination as a will be the same as that of section meant to store packets with destination as d. As a result, the section for packets with destination a will never be full and the section meant for d will always be full and the packets will be dropped even when the input buffer is not actually full. The problem is just that the empty space belongs to some other section. According to the new variation, the size will be assigned to particular section, meant for a destination a based on the percentage of packets that have this core as destination. So if the packets with destination d are more than all other packets then they will be assigned a bigger section. As there is a separate section in the input buffer for packets having each core as destination, the bullying as mentioned above is avoided even now. But along with this, the number of packets dropped can also be reduced as the size of the section meant for packets that form the majority will be greater. In this manner, the input buffers can be used optimally.

We have considered the different ways in which we can achieve better performance by modifying the design of the components. Now we can consider the method of routing. The basic method explained above has unidirectional routing. This means, the packets will be transferred from one ring register to the other only in one direction. Let us assume the order of passing is from register 1 to register 2 and so on till register 5 and register 5 to register 1. Thus if a packet at register 1 has core 5 as destination, it will have to travel to register 2, 3, 4 and only then it can reach register 5. If we consider bidirectional routing, then the same packet will just be one hop away from destination. Thus the time required to reach the destination will be less and as a result, the speed of packet flow also increases. This can lead to a reduction in the packet drop count.

We can only assume that the modifications will lead to better performance. The actual performance also depends on the traffic. We have considered a few traffic patterns and tried different combinations of the previously mentioned modifications in routing, design of the system. We have simulated the different traffic patterns with the combinations and observed the packet drop count. The different traffic patterns considered are based on the assumption that there are 5 cores and each core will be creating packets that consist of a message, a source identifier and a destination identifier. The packets are assumed to be unicast. So a packet will have only one core as destination.
First traffic pattern considered is as follows. The destination of a packet generated at a core can be any of the other cores with equal probability. The destination of the packet is chosen randomly. Here the assumption is that there is no predefined information regarding most of the packets being directed to a specific core [13].

The second pattern is such that the probability of a newly generated packet having a core c as destination is greater than that of any other core. This means most of the packets generated at a core will have a particular node as destination. Here the destination is chosen randomly again but the randomness is such that a higher weightage is given to one of the cores being chosen. This pattern is called 1-Hotspot traffic pattern.

<table>
<thead>
<tr>
<th>Buffer Design &amp; Routing</th>
<th>Number of Packets Created</th>
<th>Number of Packet Drops</th>
<th>% of Packets Drops</th>
<th>Fairness Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-1</td>
<td>456</td>
<td>245</td>
<td>53</td>
<td>0.529</td>
</tr>
<tr>
<td>Model-2</td>
<td>456</td>
<td>134</td>
<td>29</td>
<td>0.970</td>
</tr>
<tr>
<td>Model-4</td>
<td>456</td>
<td>37</td>
<td>8</td>
<td>0.960</td>
</tr>
</tbody>
</table>

TABLE I. Random distribution of traffic

The third pattern is similar to the second pattern with a small modification. Here the difference is that in this pattern, two of the cores have a higher probability of being the destination of any newly generated packet. This pattern is called 2-Hotspot traffic pattern [14].

IV. EXPERIMENT AND RESULTS

We have considered a register ring of size 5 for the simulations. We have used SystemC for the simulations. The ideas mentioned in the previous section have been used to have different combinations of buffer design and routing methods. The previously mentioned traffic patterns have been used for all the combinations. The details of the simulations have been explained here. Same traffic is created for all design and routing methods for a given distribution and the simulation is carried for equal duration.

We have considered two parameters to evaluate the effectiveness of the algorithm. They are packet drop rate and Jain’s Fairness Index. The values of these parameters have been shown below. The cost for the construction of the designs is different. So, even if a design performs better than all the others in a given scenario, the cost incurred in the construction of the design might be higher as it has to facilitate additional components like buffers, extra registers in the case of bidirectional routing, etc.

Here unidirectional routing without buffers is represented as model-1, unidirectional routing with uniform buffers is represented as model-2, unidirectional routing with varied buffer is represented by model-3, bidirectional routing with uniform buffer is represented by model-4 and bidirectional routing with varied buffer is represented by model-5.

1. Random Distribution:

This distribution involves the case where the destination core for each packet generated at each core is chosen randomly. Thus the probability of choosing any of the cores as destination is equal. The results have been shown below. Varied buffer size allocation is done based on the traffic. In this case, the traffic distribution is totally random and there is no core that will be the destination for majority of packets. As a result, varied buffer allocation will not be feasible in this situation. So only the uniform buffer has been considered in this scenario.

We can see from Table 1, that the number of packets dropped drastically comes down as we change from unidirectional routing without buffer to bidirectional routing with buffer. The percentage of dropped packets is very less in the last case.

![Comparison of Packet Drop for Random Distribution](image)

Fig 6: Bar graph showing comparison of packet drop for random distribution
2. 1-Hotspot Distribution:

In this distribution the probability of a newly generated packet having core 1 as the destination is 0.8 and the probability of the packet having any of the other cores as destination is 1/20.

<table>
<thead>
<tr>
<th>Buffer Design &amp; Routing</th>
<th>Number of Packets Created</th>
<th>Number of Packet Drops</th>
<th>% of Packets Drops</th>
<th>Fairness Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-1</td>
<td>703</td>
<td>268</td>
<td>39</td>
<td>0.627</td>
</tr>
<tr>
<td>Model-2</td>
<td>703</td>
<td>252</td>
<td>35</td>
<td>0.482</td>
</tr>
<tr>
<td>Model-3</td>
<td>703</td>
<td>208</td>
<td>29</td>
<td>0.681</td>
</tr>
<tr>
<td>Model-4</td>
<td>703</td>
<td>132</td>
<td>18</td>
<td>0.612</td>
</tr>
<tr>
<td>Model-5</td>
<td>703</td>
<td>133</td>
<td>18</td>
<td>0.771</td>
</tr>
</tbody>
</table>

We can say that as we use the idea of virtual channels, they will lead to a decrease in the number of packet drops in the case of designs involving buffers. In case of unidirectional routing, making use of variable buffers seems to be a better option as it has lesser number of packet drops and also it provides higher fairness as shown in the table. Both uniform and variable buffers provide same packet drop rate in the case of bidirectional routing. But unidirectional routing has higher value of fairness.

3. 2-Hotspot Distribution

In this distribution the probability of core 1 or core 2 being the destination of a newly generated packet is 0.35 and the probability for each of the other cores being the destination is 0.1. The results are shown in Table 3.
Bidirectional routing with varied buffer has the least packet drops. The packet drop rate comes down when we compare the design without buffer with any of the designs involving buffer. The fairness also increases when we make use of buffers.

### TABLE III. 2 Hotspot traffic distribution

<table>
<thead>
<tr>
<th>Buffer Design &amp; Routing</th>
<th>Number of Packets Created</th>
<th>Number of Packet Drops</th>
<th>% of Packet Drops</th>
<th>Fairness Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-1</td>
<td>452</td>
<td>247</td>
<td>54</td>
<td>0.504</td>
</tr>
<tr>
<td>Model-2</td>
<td>452</td>
<td>174</td>
<td>38</td>
<td>0.555</td>
</tr>
<tr>
<td>Model-3</td>
<td>452</td>
<td>130</td>
<td>28</td>
<td>0.690</td>
</tr>
<tr>
<td>Model-4</td>
<td>452</td>
<td>78</td>
<td>17</td>
<td>0.691</td>
</tr>
<tr>
<td>Model-5</td>
<td>452</td>
<td>34</td>
<td>7</td>
<td>0.590</td>
</tr>
</tbody>
</table>

Just like the previous case, varied buffer provides higher fairness and lower packet drop when compared to uniform buffer in case of unidirectional routing.

Varied buffer provides a very low packet drop rate when we consider bidirectional routing. So bidirectional routing with varied buffers can be used when there is no tight constraint on the cost incurred in the implementation of the design.

### V. CONCLUSION AND FUTURE WORK

In this paper we demonstrated the importance of virtual channels in NoC and the requirement of fair allocation of resources. Through simulations of different scenarios in System C, we observed that the performance of the ring topology varies with different modifications like bidirectional ring and by the introduction of varied buffers. Different traffic patterns like uniform, 1-hotspot and 2-hotspot were used to understand which kind of design is best suited for the given traffic pattern. The performance of varied buffer was better than the uniform buffer in all cases. The observation led us to conclude that the fair distribution of resources can improve the performance more than the equal distribution of the given resources. Bidirectional routing is suitable for some traffic patterns, while unidirectional is suited for others. In our future work, we would like to find out a suitable criterion for establishing which kind of routing is suited better for the given traffic patterns.

Our ongoing work is based on improving the performance of ring topology to match the performance of that of the crossbar. We want to introduce the kind of fairness and the performance a crossbar inhibits, while using the physical resources required by that of the ring. Researchers are facing the challenge of achieving improved performance while reducing the amount of physical resources. Our aim is to contribute to this research.
REFERENCES


