ANGLE ROUTING: A FULLY ADAPTIVE PACKET ROUTING FOR NOC

Foroogh Haj aghajani 1, Ahmad Khademzadeh 2, Samira Saeidi 3 and Kambiz Badie 2

1 Computer Engineering Department of Islamic Azad University, Science and Research Branch, Tehran, Iran
2 Iran Telecom Research Branch, Tehran, Iran
3 Computer Engineering Department of Islamic Azad University, Tehran Center Branch, Tehran, Iran

ABSTRACT

The performance of network-on-chip largely depends on the underlying routing techniques. In this paper a novel fully adaptive deadlock-free packet routing algorithm for network on chip is proposed. This method which is called angle routing (AR) determines a path based on minimizing the angle between the candidate neighbouring switch, current switch and destination. Simulation results under different traffic patterns show that, as the volume traffic of the network on chip increases, our new algorithm achieves significant better average latency compared to some other deterministic and partially adaptive routing algorithms.

KEYWORDS

Network on Chip, Adaptive Routing Algorithm, Angle Routing, Average Delay

1.INTRODUCTION

With enlarging the computation-intensive applications and low-power requirements for high performance systems, the number of computing resources on a single-chip has increased, so the interconnection among resources becomes a new challenging issue. In most SoC applications, a shared bus interconnection is used. But for the complicated systems that the number of bus requesters is large and their required bandwidth for intercommunication is more than the current bus capacity, some other interconnection network must be considered [1]. Network-on-Chip (NoC) has been proposed as a solution to provide better modularity, scalability, reliability and higher bandwidth compared to bus-based communication infrastructures [2].

In [3], suitable switching and routing algorithm has been considered as one of the design space dimensions of NoCs. In fact, the network performance and power consumption have been deeply affected by that [3]. Routing algorithm, determines the path a packet takes in the network starting from the source to its destination. Connectivity, adaptivity, fault tolerance and deadlock, livelock and starvation freedom are contemplations of routing algorithms [4].
Depending on the degree of adaptiveness, routing algorithms are classified to three routing categories. A non-adaptive routing algorithm is deterministic and routes a packet from the source to the destination along a unique, predetermined path. A minimal fully adaptive routing algorithm routes all packets through any shortest paths to the destinations. A partially adaptive routing algorithm allows multiple choices for routing packets via shortest paths, but it does not allow all packets to use any shortest paths [5].

During the passing between switches in a 2D mesh, a packet can follow four directions: east, west, north, and south. Eight distinct turns are possible in the path followed by a packet [6], [7], as shown in Figure 1.

![Figure 1. Abstract cycles in 2-D mesh [7]](image)

Algorithms with no restrictions on turns are named fully adaptive. Fully adaptive routing algorithms are subject to deadlock conditions [6].

This paper presents a fully adaptive deadlock-free routing algorithm for NoC which optimizes average latency considerably. The rest of this paper is organized as follows. Section two deals with the previous works. Within section three, the Angle routing algorithm will be discussed. In four, simulation results are presented and conclusion will be in section five.

2. RELATED WORKS

Most of the presented algorithms prohibit two or more turns to prevent deadlock, so the degree of adaptiveness decreases. [5], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], and [18] propose some of these algorithms.

Since adaptiveness increases the chances that packets may avoid hot spots or faulty components and reduces the chances that packets are continuously blocked, it can then be viewed as an important factor for routing [5]. So it is better to prevent deadlock without prohibiting turns.

Several researchers used virtual channels to design partially adaptive and fully adaptive routing algorithms for a variety of network architectures, including meshes [7], [19], [20], [21], [22], and [23]. Adding virtual channels allows the design of highly adaptive routing algorithms. However, adding virtual channels to meshes leads to a high overhead which is not favorable. It involves adding buffer space and complex control logic to routers, thus communication performance of the network and reliability of the routers may be affected [5].

In this paper, we propose a novel fully adaptive, deadlock-free routing algorithm without adding virtual channels to network on chip.
3. ANGLE ROUTING ALGORITHM

As it was mentioned, network performance has been deeply affected by routing algorithm. Here we introduce one routing algorithm that is fully adaptive and can turn to all directions. If there is only one path with healthy links and sufficient free bandwidth between source and destination, this algorithm can find it.

In this new approach priority of selecting outputs is based on the angle between destination switch, current switch and each neighbouring switch. Figures 2(a) and 2(b) show the basic concept of angle routing. In this issue, simple 2-D mesh is used. For more clarification suppose that we want to determine the priority of selecting output in figure 2(b). First the angles for all neighbouring switches of source are considered by computing $\frac{\Delta y}{\Delta x}$ and calculating absolute value of differences between it and $\frac{\pi}{2}$, $\pi$, $\frac{3\pi}{2}$ and $2\pi$. Then they will be arranged from minimum to maximum. This is going to be the priority of outputs. Therefore, the switch with maximum priority (in fact minimum angle) is selected, if the relative link is not occupied or faulty, as shown in Figure 2(b). This step is repeated for selected switch until it gets to destination. If relative link to selected switch is occupied or faulty, another neighbouring switch with higher priority (less angle) is selected.

![Figure 2. (a) Angle computation, (b) Priority of selecting next switch in a 2-D mesh](image)

However, as far as our goal is determining priority only for four neighbouring switches for simple 2-D mesh, it does not seem necessary to calculate all these angles. Indeed the priority can be considered just with being inspired by mentioned angles. For instance, if both $\Delta x$ and $\Delta y$ are positive and $\Delta y$ is more than $\Delta x$ therefore, $\theta$ is between $\frac{\pi}{2}$ and $\frac{\pi}{2}$ and the priority of switches is respectively: switch 2, switch 1, switch 3 and switch 4 which means select(Y+, X+, X-, Y-), as shown in Figure 2(b). As a result without any need for calculating angles and just with being inspired by mentioned angles, the priority of outputs can be determined. In this manner before selecting next switch, the algorithm checks path for preventing loop.

As a matter of fact since proposed algorithm is fully adaptive some techniques must be planned to avoid deadlock, livelock and starvation.

As it is mentioned in [7] there are three ways for deadlock handling including: deadlock prevention, deadlock avoidance and deadlock recovery. Deadlock prevention strategies are very conservative and they may lead to low resource utilization because of reserving all the required resources before starting packet transmission. However, deadlock avoidance and recovery
techniques evolved during the last few years significantly. Suppose a set of packets which each of them has reserved a channel when it requests another channel that is held by other packets in the set. Clearly this situation will last forever since required channel cannot be granted. Therefore, removing cyclic dependencies between channels is essential for preventing deadlock. Concerning adaptive routing since packets usually have several choices even if one of them is held by another packet, other choices may be available. Thus eliminating all the cyclic dependencies is not necessary provided that every packet can always find a path toward its destination whose channels are not involved in cyclic dependencies [7]. As proposed algorithm is fully adaptive even if just one of the channels is not occupied or faulty, packet has the permission to choose it. As a result, if there is any cyclic dependency it will be eliminated.

Respecting avoiding livelock three strategies are suggested [7]: Using only minimal paths, restricting nonminimal paths and probabilistic avoidance. Utilizing just minimal paths can prevent livelock easily but it can cause deadlock conditions and less fault tolerance. However, even when nonminimal paths are used livelock can be avoided by limiting the number of misrouting operations [7]. In our algorithm, the number of routers in packet’s path is considered for preventing livelock. In point of fact this method starts by minimal paths and if it finds reaching to destination through them implausible then it checks nonminimal paths respectively by observing loop prevention.

In regard to starvation avoidance AR focuses on transmitting packets with more delay. Delay is the absolute difference between current time and the time that packet is injected. Thus, packets with more delay have more precedence for selecting output and it leads to starvation avoidance. As for fault tolerance due to mentioned reasons and being fully adaptive, for every pair of source and destination if there is even just one path available AR is able to find it.

TO be brief, for deadlock, livelock and starvation prevention as well as fault tolerance AR:

- Eliminates cyclic dependencies
- Avoids repetitious selections
- Considers number of routers in packet’s path
- Uses precedence for packets

By using this method the average latency is meaningfully decreased because packets are not kept waiting for a long time. Additionally it always intends to find minimum feasible deviation from direct path to destination. Consequently, the way which is traversed by packets is as short as possible.

In spite of the fact that angle routing tries to find the shortest possible path to destination, but sometimes it is forced to select other paths due to occupied or faulty links. Hence in these situations packets consume more energy because of passing longer paths. To solve this problem packet’s delay is also considered for selecting output. As it was mentioned, delay is the absolute difference between current time and the time that packet is injected. It is clear that for those packets which are not involved in long delay there is no need for making the path longer and consuming more energy. Therefore, angle routing is just used for the packets whose delay is more than d. In fact d is a parameter which determines the degree of adaptiveness of routing algorithm. Actually the number of packets which use angle routing is influenced by d directly. Consequently, it can be seen that the more d is, the less adaptiveness the routing algorithm has. Therefore, the percentage of packets which use angle routing is limited to the packets with long delay. It causes considerable reduction in average delay because these packets are the main
reason for increasing average delay. Moreover, if $d$ is properly set the percentage of these packets is not so much that causes any noticeable effect on energy consumption.

Furthermore, the number of routers in packet's path (hop number) is considered for livelock avoidance. In fact angle routing is used just for the packets which their hop number is less than specific value. Clearly it should be proportional to the distance between source and destination. In presented algorithm $m*\text{InitialDistance}$ is used to show maximum allowable hop number which $\text{InitialDistance}$ is sum of horizontal and vertical distances between source and destination and $m$ is another factor which affects adaptiveness directly. As a matter of fact the more $m$ is, the more adaptive the routing algorithm.

The pseudo code of this algorithm for 2-d mesh is brought in below.

**Algorithm: Angle Routing Algorithm for 2-D Meshes**

**Inputs:** Coordinates of current node $(X_{\text{current}}, Y_{\text{current}})$, 
Source node $(X_{\text{src}}, Y_{\text{src}})$, 
Destination node $(X_{\text{dest}}, Y_{\text{dest}})$, 
Packet injection time $(\text{PacketInjectionTime})$, 
Number of routers (hop number) / * Number of Routers in Packet’s Path up to Current Node*/

**Output:** Selected output Channel

**Procedure:** /*(this procedure is executed for the packet with maximum precedence (delay) in order to starvation prevention)*/

\[
\text{Xoffset := } X_{\text{dest}} - X_{\text{current}}; \\
\text{Yoffset := } Y_{\text{dest}} - Y_{\text{current}}; \\
\text{InitialDistance := abs(Xoffset) + abs(Yoffset);} \\
\text{Delay := } \text{CurrentTime} - \text{PacketInjectionTime}; \\
\text{if Xoffset = 0 and Yoffset = 0 then} \\
\text{Channel := Internal;} \\
\text{else if (Delay} \leq d \text{ or RouterNumber} \geq m* \text{InitialDistance) then} \\
\text{Return RoutingXY (Xcurrent, Ycurrent, Xdes, Ydes);} \\
\text{else} \\
\text{if Xoffset > 0 and Yoffset >= 0 then} \\
\text{if Xoffset > Yoffset then} \\
\text{Channel := Select(X+, Y+, Y-, X-);} \\
\text{else} \\
\text{Channel := Select(Y+, X+, X-, Y-);} \\
\text{endif} \\
\text{else if Xoffset <= 0 and Yoffset > 0 then} \\
\text{if Yoffset > Xoffset then /*absolute(Xoffset) > absolute(Yoffset)*/} \\
\text{Channel := Select(X-, Y+, Y-, X+);} \\
\text{else} \\
\text{Channel := Select(Y+, X-, X+, Y-);} \\
\text{endif} \\
\text{else if Xoffset < 0 and Yoffset <= 0 then} \\
\text{if Xoffset < Yoffset then /*absolute(Xoffset) > absolute(Yoffset)*/} \\
\text{Channel := Select(X-, Y-, Y+, X+);} \\
\text{else} \\
\text{Channel := Select(Y-, X-, X+, Y+);} \\
\text{endif} \\
\text{else if Xoffset >= 0 and Yoffset < 0 then} \\
\text{if Yoffset > Xoffset then /*absolute(Xoffset) > absolute(Yoffset)*/} \\
\text{Channel := Select(X+, Y-, Y+, X-);} \\
\text{else} \\
\text{Channel := Select(Y-, X+, X-, Y+);} \\
\text{endif} \\
\text{else} \\
\text{Channel := Select(Y-, X+, X-, Y+);} \]
In above algorithm, according to acquired priority, output will be selected if the relative link is not occupied or faulty. In addition for preventing repetition, it avoids selecting the same direction that packet comes from.

As it can be seen in the results, this method presents better average latency in comparison with other considered algorithms.

For comparing behavior of this algorithm with other algorithms, routing is done for a packet in spite of occupied or faulty links by various ways of routing in Figure 3. The intent of occupied links is the links which other applications have higher precedence to use them during the time that this packet should get to destination.

In these Figures, the circle with S is the source of packet, the circle with D is the destination of packet, the dashed links show occupied or faulty links and the dark links show selected links in each routing algorithm.

In Figure 3(a), routing is done by the x-y routing method. As we know x-y routing is a deterministic technique, hence as soon as a packet reaches an occupied or faulty link, it is compelled to stop. As shown in Figure 3(a), the packet stops and cannot get to the destination because the dashed link with a × symbol on it, is occupied or faulty.

If we use west first, north last and negative first algorithms like in Figures 3(b)-(d) for the same situation data could not reach destination because of prohibiting some turns.
But by the proposed algorithm, data could get to destination as shown in Figure 4, for the reason of not prohibiting any turn for preventing deadlock.

As it was mentioned due to acquire better performance in regard to energy consumption, angle routing is just used for some packets and for the rest of packets XY routing algorithm is used. In view of the fact that regarding average latency under some pattern traffics, other partially adaptive routing algorithms such as west-first, negative-first and north-last perform better in comparison with XY, they can be used in our algorithm instead of XY. In fact if each of mentioned algorithms is used in place of XY, it will lead to a combination of these algorithms and angle routing algorithm. For instance, if negative-first is used instead of XY, we will have angle-negative-first (ANF) algorithm and so on. However, for each case average packet latency decreases since packets mostly are not kept waiting for a long time but average energy consumption increases because some packets are compelled to pass the longer path.
4. ANALYSIS OF SIMULATION RESULTS

In order to assess the performance of the purposed algorithm, we simulated it together with other routing techniques by using a systemC based simulator which is called NOXIM [24] and compared their results. Obviously, selecting suitable criteria to survey received consequences is a serious subject. In this work, the efficiency of each type of routing is evaluated through latency-traffic and energy-traffic curves. It is assumed that the packet latency spans instantly when the first flit of the packet is created, to the time when the last flit is ejected to the destination node, including the queuing time at the source. As for energy, the average energy which is consumed by all received packets is considered including the required energy for forwarding flits, incoming flits, flits which are standby and the energy which is used for performing routing algorithm.

Each simulation is run for a warm-up period of 1000 cycles. Thereafter, performance data are collected for 10000 cycles. This procedure is repeated for 1000 times and presented results are the averages of all collected data.

The network size during simulation is fixed to be 7*7 tiles. Each channel of all routers has a FIFO buffer size of 4 flits. The minimum and maximum of packets’ sizes are 2 and 10 flits respectively. Since the network performance is significantly influenced by the traffic pattern, three traffic patterns are considered: uniform, transpose and hot spot. In the uniform traffic pattern, each core sends a packet to any other cores with equal probability. For the transpose traffic, two types of patterns are simulated. With the first transpose traffic pattern a core at (i, j) only sends packets to the core at (6-j, 6-i). In the second transpose traffic pattern a core at (i, j) only sends packets to the core at (j, i). In the hot spot traffic pattern, the cores at (2, 2) and (5, 5) are designated as the hot spots, which receive 10% more traffic in addition to the regular uniform traffic.

As it was mentioned in order to achieve better performance regarding energy consumption, angle routing is just used for packets whose delay is higher than \( d \) and for the rest of packets other deterministic or partially adaptive algorithms such as XY, negative first, west first and north last are used. Therefore as \( d \) decreases the percentage of packets which use angle routing increases and as a result the number of packets kept waiting falls and it leads to better average packet latency. However, it causes the energy consumption to go up since more packets pass longer paths. In fact the new algorithm inherits some features like average latency and average energy to some extent. For instance, concerning angle-negative-first (ANF) due to mentioned reasons we have less average latency and more average energy consumption compared with negative first.

In Figure 5 the results of negative first and ANF for various values of \( d \) under the second transpose traffic are displayed. As it can be seen in Figure 5(a) by reducing \( d \), average packet latency decreases significantly compared to slight increment of energy consumption which is shown in Figure 5(b). For instance, concerning ANF 4000 for packet injection rate over .06 (packets/cycle) average packet latency decreases by more than 50% compared with NF while energy consumption faces just less than 10% increase. Even if we consider greater values of \( d \), it is observed that in spite of trifling growth of energy consumption, improvement of average packet latency is meaningful. However, \( d \) can be changed for different systems. When smaller \( d \) is considered, betterment of average latency is even greater but average energy consumption could face considerable increments.
As it was noted, traffic pattern deeply affects the network performance. Figures 6-8 show the simulation results for various kinds of traffic. In Figures 6(a) and 6(b), average packet latency and average packet energy under uniform traffic for two values of d 2000 and 4500 are studied. As expected, for uniform traffic XY presents better performance compared with partially adaptive algorithms like west first and north last. However, owing to the reasons mentioned before, angle-XY (AXY), angle-west-first (AWF) and angle-north-last (ANL) offers better average packet latency but more average packet energy in comparison with XY, west first and north last.

Figures 7(a) and 7(b) compare performance of algorithms under the first transpose traffic as well as Figures 7(c) and 7(d) which represent average packet latency and average packet energy for the second transpose traffic for two values of d 1000 and 6000. Taking the results using transpose traffic, improvement of average latency for all algorithms which use angle routing is of a considerable amount compared to additional power overheads. As shown in Figures 7(a)-(d) since adaptive algorithms naturally behave better under non uniform traffics, performance gains of our algorithm for transpose traffic ameliorate significantly contrasted with uniform traffic. Especially as Figure 7(c) reveals for smaller delays our algorithm saturates at considerable high injection in comparison with other considered algorithms.
Figure 7 (a) Average packet latency under the first transpose traffic, (b) Average packet Energy under the first transpose traffic, (c) Average packet latency under the second transpose traffic, (d) Average packet Energy under the second transpose traffic.

In Figures 8(a) and 8(b), we show simulation results of algorithms under hot spot traffic for two values of \( d \) 2000 and 6000. Here hot spot traffic with two hot spot nodes, which receive 10% more traffic extra to the regular uniform traffic, is considered. Figures 8(a) and 8(b) represent that on one hand angle routing optimizes average delay, but on the other hand it augments average energy noticeably in compassion with transpose traffic.

Figure 8. (a) Average packet latency under hot spot traffic, (b) Average packet Energy under hot spot traffic.
On the whole our observation of the simulation results can be summarized as follows: Although XY and partially adaptive algorithms perform better in regard to average energy consumption, our algorithm presents significant improvement concerning average packet latency. Another interesting fact is that angle routing algorithm does keep the advantages of deterministic and partially adaptive routing when network is not congested. Furthermore, depends on the priority of our needs the degree of adaptiveness can be changed. It means whenever average delay is more important, smaller values of d is designated, otherwise provided that average energy is intended to be more prominent, greater values of d would be preferred.

5. CONCLUSION

In this paper, a novel fully adaptive deadlock-free packet routing algorithm is presented, based on being inspired by angle computation in 2-D meshes for Network on Chip without adding any virtual channels.

As we know, decreasing average latency is one of the most significant objectives in routing algorithms. According to the simulation results, average packet latency appears meaningfully better in angle routing compared to other considered algorithms. In addition, as the priority is computed in each step of routing, the proposed method was realized to be suitable for dynamic applications. Moreover, using AR for three dimensional networks is an interesting subject for future works.

REFERENCES


