计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150149
53(6):1211-1220,2016

众核处理器片上网络的层次化全局自适应路由机制

张 洋[1,2] 王 达[1] 叶笑春[1] 朱亚涛[1,2,3] 范东睿[1] 李宏亮[4] 谢向辉[4]
[1](计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)
[2](中国科学院大学计算机与控制学院 北京 100049)
[3](河北农业大学信息科学与技术学院 河北保定 071001)
[4](数学工程与先进计算国家重点实验室 江苏无锡 214125)
(zhangyang@ict.ac.cn)

A Global Hierarchical Adaptive Routing Mechanism in Many-Core Processor Network-on-Chip

Zhang Yang[1,2], Wang Da[1], Ye Xiaochun[1], Zhu Yatao[1,2,3], Fan Dongrui[1], Li Hongliang[4], and Xie Xianghui[4]
[1](*State Key Laboratory of Computer Architecture* (*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190)
[2](*School of Computer and Control Engineering*, *University of Chinese Academy of Sciences*, *Beijing* 100049)
[3](*College of Information Science & Technology*, *Agricultural University of Hebei*, *Baoding*, *Hebei* 071001)
[4](*State Key Laboratory of Mathematical Engineering and Advanced Computing*, *Wuxi*, *Jiangsu* 214125)

Abstract Accompanied by the arrival of the era of big data, data center has been becoming an infrastructure in human life. Many-core processor provides a highly parallel capability to solve applications in data center such as sorting and searching efficiently. For the purpose to utilize the parallelism of many-core processor, routing algorithm in interconnection network turns into one of the most important issues in many-core system. Mesh and ring are the most employed topological structures in many-core processor for their features of easy implementation and high scalability. Depending on the scope of congestion information, routing algorithms in mesh and ring can be divided into oblivious routing, local adaptive routing, regional adaptive routing and global adaptive routing. The oblivious routing algorithm applied in the mesh architecture affects the load-balance of the network which is reflected in reducing through-put and high packet latency. Current local adaptive routing and regional adaptive routing both suffer from short-sightedness and are not suitable for large scale mesh structure. And prior global adaptive routings are limited by the slow computation of global route. We propose a novel global hierarchical adaptive routing mechanism, which is comprised of a global congestion information propagation network Roof-Mesh and a global hierarchical adaptive routing algorithm GHARA. Roof-Mesh provides a platform to share global congestion information in

收稿日期:2015-02-15;**修回日期**:2015-07-14
基金项目:国家"九七三"重点基础研究发展计划基金项目(2011CB302501);国家自然科学基金项目(61332009,61173007,61221062);"核高基"国家科技重大专项基金项目(2013ZX0102-8001-001-001);国家"八六三"高技术研究发展计划基金项目(2015AA011204,2012AA010901)
This work was supported by the National Basic Research Program of China (973 Program) (2011CB302501), the National Natural Science Foundation of China (61332009,61173007,61221062), the National Science and Technology Major Projects of Hegaoji (2013ZX0102-8001-001-001), and the National High Technology Research and Development Program of China (863 Program) (2015AA011204,2012AA010901).

a hierarchical way among all nodes on the network. Depending on the information supplied by Roof-Mesh, GHARA reduces the procedure of routing by hierarchically computing from large region perspective to neighbor nodes. The result of experiment shows that GHARA performs better than other regional and global adaptive routings.

Key words many-core processor; networks-on-chip; load balance; global congestion information propagation network; global hierarchical adaptive routing algorithm (GHARA); Roof-Mesh

摘 要 Mesh和环拓扑结构以其实现简单、易于扩展的特点成为众核处理器片上网络应用最为广泛的拓扑结构.应用于Mesh结构中的健忘型路由算法在网络流量较大时影响片上网络的负载均衡,表现在降低吞吐量和增大数据包延迟.自适应算法中的本地自适应算法和区域自适应算法均存在不同程度的短视现象,不适合大规模的Mesh结构,而目前全局自适应算法又由于路由计算量大而速度缓慢.提出一种新的层次化全局自适应路由机制,包括一个全局拥塞信息传播网络Roof-Mesh和一个层次化全局自适应路由算法(global hierarchical adaptive routing algorithm, GHARA).通过全局拥塞信息传播网络得到拥塞信息,GHARA采用全网分区逐级计算路由的方式,减少了全局路由的计算步骤,从而减少了平均数据包延迟、提升了饱和带宽.实验结果表明GHARA表现优于其他区域和全局自适应路由算法.在人工注入通信模式下,8×8 Mesh平均饱和带宽比全局自适应算法GCA提高10.7%,16×16 Mesh平均饱和带宽比全局自适应算法GCA提高14.7%.在运行真实测试程序集SPLASH-2模式下,数据包延迟最高比GCA提高40%,平均提升14%.

关键词 众核处理器;片上网络;负载均衡;全局拥塞信息传播网络;层次化全局自适应路由算法;Roof-Mesh

中图法分类号 TP302

多核和众核处理器已成为业界的主流.在多核、众核处理器以及片上系统的设计中,片上网络互联结构以其高带宽、低延迟、扩展性强,容错性好等特点迅速取代了共享总线等其他片上连接模式,成为该领域的主流互联方法.迄今为止,大部分片上网络的拓扑结构的研究借鉴了并行计算体系结构中的网络结构,而在实际应用中,Mesh[1-3]由于组成简单、连线较短的特点,逐渐成为使用结构最为广泛的结构.由于在此类拓扑结构中,从源节点到目的节点存在多条路径,因此片上网络的路由算法需要在众多路径中做出选择.路由算法将影响整个网络的负载均衡,具体体现在延迟和吞吐量的变化.

根据路由依据条件的不同,片上网络的路由算法可简单地分为健忘性算法和自适应算法.健忘性算法如维序路由(dimension ordered routing, DOR)在路由仲裁过程中完全不考虑网络的拥塞状态,只根据源节点和目的节点的位置确定路由.尽管健忘性算法的复杂度低,但是在网络流量较大时却会造成严重的拥塞而使得网络负载不均衡[4].相反地,自适应算法根据网络拥塞状态,在源-目的节点之间动态选择相对不拥塞的1条路径来平衡网络负载,较之健忘性负载,自适应算法能带来更好的片上网络性能,逐渐成为大规模片上网络主流路由算法.但是自适应算法很难有效地获得全局负载信息,即使一些算法可以获得这部分信息,因为全局信息数据量庞大也影响到了计算路由的速度.

在自适应算法的实现过程中,不同的算法获得拥塞信息的方法不同.依据获得网络节点信息规模的大小,可分为本地自适应(local adaptive)算法、区域自适应(regional adaptive)算法和全局自适应(global adaptive)算法[5].本地自适应算法用本地拥塞信息来决定路由,实现起来比较简单.但是本地自适应算法会因为前期的短视现象导致后期不得不选择高拥塞路径.和本地自适应相比,区域自适应算法扩大了自适应范围.但是由于依旧存在一定范围的限制,仲裁出的路由仍然有可能将通信引入自适应范围之外的拥塞区.全局自适应算法能感知全局拥塞信息,进而指导路由,能最大程度避免负载均衡.但是目前已知的全局路由算法均存在不同程度的更新负载信息或者计算路由较慢的问题.

针对本地自适应算法和区域自适应算法很难有效获得全局负载信息同时全局自适应算法计算路由

速度慢的问题,本文提出一种全局拥塞信息传播结构 Roof-Mesh,同时基于这种 Roof-Mesh 结构提出一种层次化全局自适应路由算法(global hierarchical adaptive routing algorithm, GHARA).本文的主要贡献为:

1) 提出的全局拥塞信息传播结构 Roof-Mesh 将 $N\times N$ 的 Mesh 网络分为 $\mathrm{lb}\,N-1$ 级子网络,每个子网都有一个附加网络进行子网节点的拥塞信息的收集和传播.每一级附加网络对其下一级附加网络的拥塞信息进行处理.

2) 本文基于 Roof-Mesh 提出一种混合精确粒度的层次化全局自适应路由算法 GHARA.实验结果表明 GHARA 表现优于其他区域和全局自适应路由算法.在人工注入通信模式下,8×8 Mesh 的平均饱和带宽比全局拥塞感知(global congestion awareness, GCA)提高 10.7%,16×16 Mesh 平均饱和带宽比 GCA 提高 14.7%.而在运行真实测试程序集 SPLASH-2[6] 模式下,数据包延迟最高比 GCA 提高 40%,平均提升 14%.

1　相关工作

Dally 和 Aoki[7] 用当前节点空闲虚通道(virtual channel, VC)的数目来作为拥塞指标,路由器选择拥有较多虚通道的输出端口作为当前节点输出方向.Li 等人[8] 在 DyXY 算法中,用节点的输入缓冲队列长度来表示 1 个节点的压力值,节点通过监视相邻节点的压力值来作为路由仲裁的依据.这些方法只利用本地拥塞信息来决定路由,实现起来比较简单,而且不会因为传播拥塞信息而对网络通信造成影响.但是从全局角度来看,本地自适应算法会因为前期的短视现象导致后期不得不选择高拥塞路径.在 Gratz 等人[9] 提出的区域拥塞感知(regional congestion awareness, RCA)中,首次提出监视范围超越邻居节点的路由机制.RCA 通过一个额外网络将局部拥塞信息传播给更远的节点.Ma 等人[10] 提出基于目的的自适应路由算法(destination-based adaptive routing, DBAR)改进了 RCA,使得拥塞结果更为精确.和本地自适应相比,区域自适应扩大了自适应范围,为路由仲裁提供了更好的依据.但是由于依旧存在一定范围的限制,按照拥塞结果仲裁出的路由仍然有可能将通信引入监视范围之外的拥塞区.Manevich 等人[11] 提出自适应切换维度顺序路由算法 ATDOR(adaptive toggle dimension ordered routing),在这个算法中,Manevich 等人用 1 个附加网络传送各节点拥塞信息到 1 个表中,以此来为每个源-目的节点对选择 XY 或者 YX 维序路由.这个方法得到即时全局拥塞信息非常精确,但是随着网络规模的增加,拥塞表更新的速度会变慢.在 GCA 中,Gratz 等人[5] 使用“背负式”(piggybacking)信息模式,将数据包所经过的节点的拥塞信息嵌入到数据包头中进行传播.各节点提取经过自己的数据包头中的拥塞信息来更新各自的全局拥塞信息表.这个办法的开销很小,但是拥塞信息只能被数据经过的节点感知,同时 GCA 由于计算路由的时间较长而影响了自身的性能.

本文提出一种全局拥塞信息传播结构 Roof-Mesh,以及基于 Roof-Mesh 结构的层次化全局自适应路由算法 GHARA,减少了计算步骤,从而降低了网络数据包延迟,提高了饱和带宽.

2　Roof-Mesh 片上网络

本文所提出的 Roof-Mesh 是一种多层次片上网络全局拥塞信息传播结构.理论上这种结构可以监听传播所有片上网络拓扑结构的拥塞信息,本文只针对 Mesh 这种结构的设计展开.为了便于描述,本文以一个 8×8 的 Mesh 网络为例进行介绍.本文将在第 4 节中对在更大规模的网络上应用基于该结构的算法进行评估.

2.1　Roof-Mesh 的基本组成

在 Roof-Mesh 结构中,节点的拥塞信息通过节点 router 内各方向端口可用的输入 VC 数衡量,1 个方向上可用 VC 数越少,表示这个方向上拥塞程度等级越高.拥塞程度可分为多个不同等级,每个等级对应一组可用 VC 数量.VC 数量上细微的差别对于网络通信的影响不明显[10],为了节省传递拥塞信息的附加网络的带宽,在本文的设计中我们将拥塞程度分成 2 个等级:拥塞和不拥塞,可用 VC 数量少于 VC 总数的一半表示这个端口拥塞,反之可用 VC 数量超过 VC 总数的一半表示不拥塞.按照分层次拥塞传播的思路,每 4 个节点组成 1 个 G4 组,4 个 G4 组组成 1 个 G16 组.组拥塞状态也分为类似开关的 2 个等级,以节省信息传播开销.每个节点 router 连接到 1 级附加网络中的上行带宽为 4 b,为 1 组 4 个方向拥塞信息的等级值.下行 5 b,其中 2 b 为拥塞信息类型,3 b 为 1 组 3 个拥塞信息的等级值.对于 1 个 8×8 的 Mesh 结构来说,拥塞信息类型可

分类为每个节点的拥塞信息,每 4 个节点组成的 G4 组拥塞信息和每 16 点组成的 G16 组拥塞信息. 节点 router 上传自己 4 个方向的拥塞信息,接收同在 1 个 G4 组内的其他 3 个节点的拥塞信息, 同在一个 G16 组内的其他 3 个 G4 组的拥塞信息以及其他 3 个 G16 的拥塞信息.

2.2 多级拥塞传播附加网络

每 16 个节点组成的 G16 组,构成 1 个 1 级拥塞传播附加网络,如图 1 所示. G16 内的所有节点将自己的拥塞信息传给附加网络中央的 1 级拥塞管理单元(CMU-L1),CMU-L1 存储 G16 的所有节点详细拥塞等级信息,根据各个节点的位置,CMU-L1 返回给各个节点其他节点相应方向端口的拥塞信息以及由这些信息组成的 4 个 G4 组信息. CMU-L1 的结构如图 2 所示,16 个节点将自己的拥塞信息传入 CMU-L1 中的组内节点拥塞信息寄存器,节点拥塞寄存器将内容交予拥塞信息处理单元处理. 处理工作包含将节点信息整合成 G4 组拥塞信息和 G16 组拥塞信息,以及将不同方向的节点拥塞信息分类传给不同位置的节点. 对于 1 个节点来说,其他节点哪个方向的拥塞值对他来说有意义取决于 2 个节点的位置. 节点所在 X 和 Y 维将网络分为 4 个象限,对于不同象限来说,节点关注它们不同方向的拥塞值.

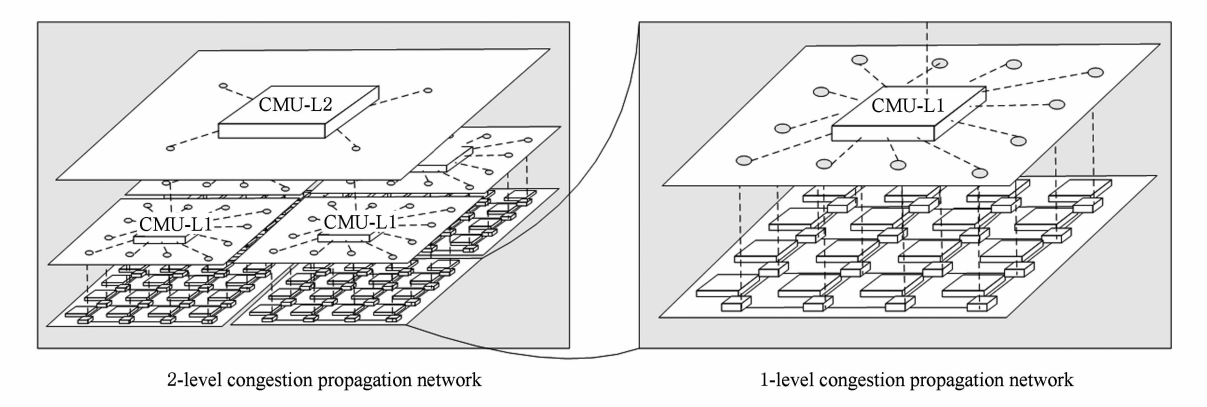


Fig. 1 Roof-Mesh network.

图 1 Roof-Mesh 网络

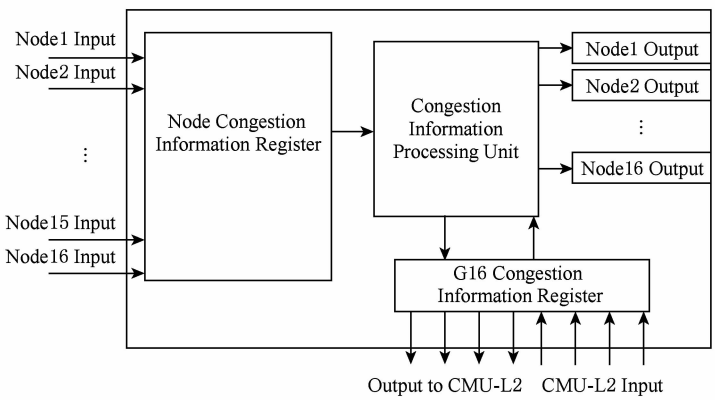


Fig. 2 CMU-L1 micro-architecture.

图 2 CMU-L1 微结构

4 个 G16 组成 1 个 G64,每个 G16 的 CMU-L1 连接至 1 个 2 级拥塞监控单元(CMU-L2), 构成 1 个 2 级拥塞监控网络,如图 1 左图所示. CMU-L2 的作用是收集并传播 G16 的拥塞信息,由于 G16 在第 3 节提到的路由算法中属于相对远距离参考因素,因此我们仅收集和传播 G16 组拥塞程度的加权平均信息. 由于处于 2 个 G16 边界的拥塞相对于全网边缘的拥塞对路由的影响更大,我们在处理计算 G16 组平均拥塞时,对 2 个 G16 边界的拥塞增加了 1 倍的权重. CMU-L2 的结构如图 3 所示. 在 8×8 的 Mesh 结构中,2 级拥塞传播网络即为顶层网络. 在顶层网络的 CMU 中,拥塞信息处理单元将 4 组 G16 的拥塞信息分类传回低一级 CMU 中.

图 4 表示在每个节点路由器内的拥塞信息寄存

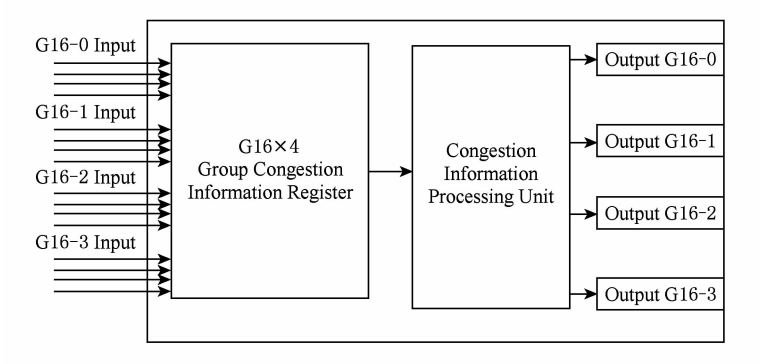


Fig. 3　CMU-L2 micro-architecture.

图 3　CMU-L2 微结构

器存储了和此节点同一个 G4 组内的其他节点的拥塞信息、此 G4 组同一个 G16 组内其他 G4 组的拥塞信息以及其他 G16 组的拥塞信息. 其中相邻的节点、G4 组或 G16 组拥塞信息存储整体平均拥塞值,不相邻的则存储 2 个方向单独的平均值.

E	W	S	N	CN-1	CN-3	CN-2	CN-2
C4-1	C4-3	C4-2	C4-2	C16-1	C16-3	C16-2	C16-2

E, W, S, N: four-directional congestion of a node;
CN-1: congestion of the first clockwise node in G4;
C4-1: congestion of the first clockwise G4 in G16;
C16-1: congestion of the first clockwise G16.

Fig. 4　Congestion information register.

图 4　拥塞信息寄存器

3　层次化全局自适应路由算法

基于全局范围内的拥塞信息,我们提出一种层次化全局自适应路由算法 GHARA. 这种算法根据每个路由器从拥塞监控网络得到即时的全网络范围内的拥塞状态信息,选择合适的输出端口. 由于该算法参考的信息为全局拥塞状态信息,因此解决了本地和区域自适应路由算法的短视问题.

3.1　GHARA 算法

如图 5 所示的 G16 grouping 图,基于 Roof-Mesh 网络,将 8×8 的 Mesh 网络分割成 4 个 4×4 的区域,即第 1 节中提到的 G16. 每个 G16 分成 4 个 2×2 的区域 G4. 通过 Roof-Mesh,每个节点中的拥塞信息寄存器得到 2 个相邻节点的对应方向接口拥塞信息,以及在同一个 G4 组中对角线上节点对应 2 个方向拥塞信息. 除此之外,得到同一个 G16 组内其他 G4 组的平均拥塞信息以及其他 3 个 G16 组的平均拥塞信息. 由于较远的节点的即时拥塞信息可能在以后数据经过时发生变化,同时出于开销的考虑,在计算路由时,除了节点本身所在 G4 组内的节点拥塞值使用精确信息以外,其他相对较远的节点我们都用其组内平均拥塞信息.

GHARA 算法步骤如下:

1) GHARA 计算 2 个节点之间的路由,首先判断 2 个节点属于哪 2 个 G16 组.

2) 把 G16 组抽象成多个节点,G16 组的加权平均拥塞值视为抽象节点拥塞值. 判断源节点所在抽象

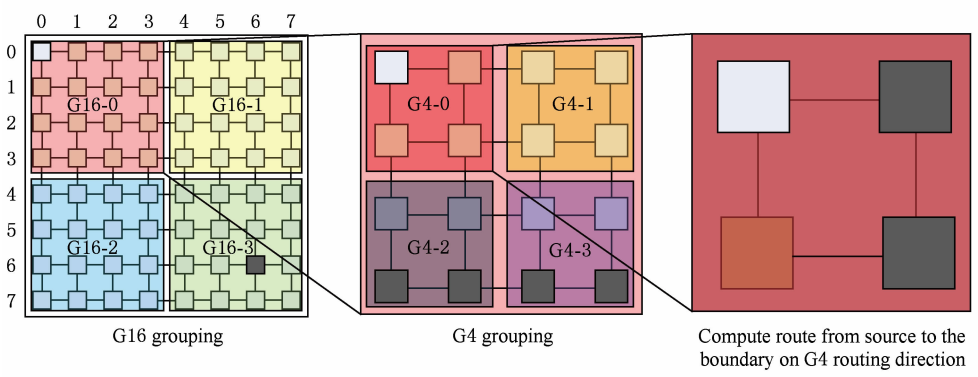


Fig. 5　Global hierarchical adaptive routing algorithm (GHARA).

图 5　层次化全局自适应路由算法

节点到目的节点所在抽象节点的路由,即 G16 组级路由.

3) 通过抽象出的 G16 组级路由,在源节点所在 G16 组内定位组级路由方向上的边界节点集合.

4) 计算从源节点至 G16 组边界节点集合的最小拥塞路径.将 G16 内的 4 个 G4 组抽象成 4 个节点,4 个 G4 组各自的加权平均拥塞值视为抽象节点拥塞值.计算这 4 个抽象节点间至边界的路由,即 G4 组级路由.

5) 通过抽象出的 G4 组级路由,在源节点所在 G4 组内定位 G4 组级路由方向上的边界节点集合.

6) 计算到边界上 2 个节点的最小拥塞路径.找到源节点的输出端口.

算法 1. GHARA 算法.

① $source=Group(source,n)$; /* n 为级数 */
② $destination=Group(destination,n)$;
③ if $source=destination.neighbour$
④ $temp=Direction(source,destination)$;
⑤ else
⑥ $temp_1=Congestion(source.clockwise)+Congestion(destination.clockwise)+Congestion(source.leftneigbour)$;
⑦ $temp_2=Congestion(source.anticlockwise)+Congestion(destination.anticlockwise)+Congestion(source.rightneigbour)$;
⑧ end if
⑨ if $temp_1<temp_2$
⑩ $temp=source.leftneigbour$;
⑪ else
⑫ $temp=source.rightneibour$;
⑬ end if
⑭ for ($i=n-1,n-2,\cdots,1$)
⑮ $source=Group(source,i)$;
⑯ $dest[]=Group\_border(temp,source,i)$;
⑰ $temp=SubRoute(source,dest[],i)$;
⑱ end for

下面举例说明 GHARA 算法.图 5 计算 1 个从(0,0)到(6,6)的路由,每一步计算路由的目标均用黑色标注.节点(0,0)所在 G16 组为 G16-0,节点(6,6)所在 G16 组为 G16-3.从拥塞信息寄存器中获得 G16-1 和 G16-2 的拥塞值分别为 1 和 0,因此路由选择从 G16-2 区域经过.在 G16-0 中,每一跳都将计算到区域 G16-0 和 G16-2 的边境点(3,0), (3,1), (3,2), (3,3)之一的最小拥塞距离.在计算 G16 内的最小拥塞路径时,我们继续将 G16 内的 G4 组看作是 4 个节点进行计算,从拥塞信息寄存器中我们获得 G4-1,G4-2,G4-3 的拥塞值分别为 0,1,0.因此选择从 G4-1,G4-3 抵达 G16-0 的边境点(3,2),(3,3).接下来在 G4-0 中要计算节点(0,0)到 G4 边境节点(0,1)和(1,1)的最小拥塞.从拥塞信息寄存器中获得节点(0,1),(1,0)的拥塞信息分别 0 和 1,同时得到节点(1,1) 2 个方向的拥塞信息为 1 和 1.因此将选择(0,1)为下一个节点,输出端口为 E.(0,1)为边境目的节点则选择过境,进入 G4-1,之后重复在 G4-0 内的计算过程,数据包进入 G4-3,选择边境目的节点,到达 G16 边境,然后进入 G16-2 开始下一轮路由计算.

本文提出的方法对比全局自适应算法 GCA,虽然牺牲了不敏感的远距离节点间的数据精度,但是将全网路由最大计算步骤从 $2\sqrt{N}-3$ 减少至 $\mathrm{lb}\,N$ 次,提高了片上网络的性能.

3.2 支持 GHARA 的路由器微结构

GHARA 的路由器如图 6 所示,我们以 Ma 等人在文献[10]提出的 VC 路由器作为基线路由器,其流水线分为路由计算(RC)、VC 分配(VA)、交换分配(SA)和交换传送(ST)这 4 级以及 1 个周期的 LT(链路传送).该路由器为了提高性能,利用 lookahead 路由模式将 RC 提前执行以缩短流水线深度[12-13].同时,在 ST 流水级所在周期编码目的节点地址信号和边境目的地址信号,同时进行下一个路由器到全网各节点的输出端口计算[14-15].在 LT 所在周期,下一个路由器通过上一周期产生的目的节点编码信号和全网各节点路由表预取即将到来 flit 的输出端口.

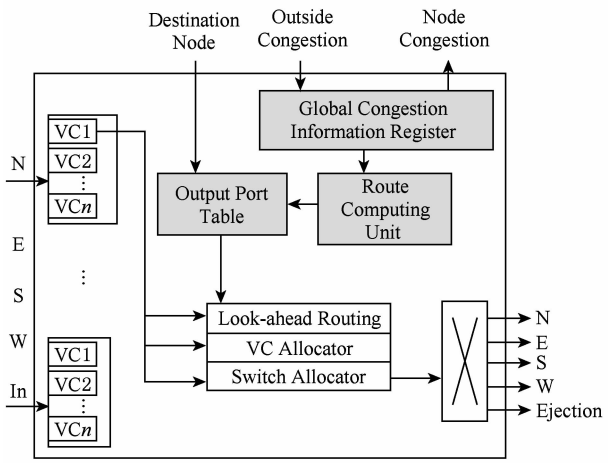


Fig. 6 GHARA router micro-architecture.

图 6 GHARA 路由器微结构

在基线处理器的基础上,我们修改了原有的 X 维和 Y 维的拥塞寄存器,添加了 1 个第 1 节提到的全局拥塞信息寄存器.拥塞信息通过监控网络存储在寄存器中,根据寄存器中的拥塞信息,路由计算单元根据拥塞信息通过 GHARA 计算当前节点到全网每个节点的路由对应输出端口.将到每个节点应选择的输出端口结果存储在输出端口表中.由目的地址的编码去选取输出端口表中的对应值.

本节详细介绍了基于 Roof-Mesh 结构的层次化全局自适应路由算法 GHARA 以及实现算法的路由器微结构.这种算法根据每个路由器从拥塞监控网络得到即时的全网络范围内的拥塞状态信息,选择合适的输出端口.由于该算法参考的信息为全局拥塞状态信息,因此解决了本地自适应路由算法的短视问题.同时,该算法将全网路由最大计算步骤从 $2\sqrt{N}-3$ 减少至 $\mathrm{lb}\,N$ 次,降低了计算路由产生的额外延迟.

4　实验评估

本节评估 GHARA 在多种人工片上通信模式以及实际负载测试程序集上的性能.

4.1　实验设置

我们修改了片上网络模拟器 booksim2.0[16] 来实现 Roof-Mesh 拥塞监控网络结构以及 GHARA 所需要的路由器微结构.我们利用 GEM5[17] 来抓取 booksim2.0 所需要格式的测试程序 trace.本文分别从确定路由算法、本地自适应路由算法、区域自适应算法和全局自适应算法中选取了 DOR,Local,RCA,GCA 作为 GHARA 的对比算法进行评估.

本文采用 8×8 和 16×16 的 2D Mesh 的拓扑结构,路由器的流水线合并了 RC 和 VC,SA 使用 2 个流水级,每一跳除了流水线中的 2 个周期,还需要 1 个周期的链接传输.每个端口上使用 8 个 VC,每个 VC 有 5 个微片缓冲.片上网络模拟的预热需要 10 000 个周期,数据采样来自于接下来的 100 000 个周期.

在评估人工注入传输模式下的片上网络性能时使用了刷洗(shuffle)、转置(transpose)和位反(bit-reverse)三种通信传输模式,在评估真实测试程序下的片上网络性能时使用了 SPLASH-2 测试用例集中的 fft,raydix,raytrace,ocean,barnes.

4.2　实验结果与分析

1) 真实测试程序.图 7 给出了 5 种算法在 8×8 Mesh 网络下运行实际科学计算测试程序 SPLASH-2 的平均数据包延迟.为了直观比较各种算法的性能,所有运行结果进行了对 DOR 的结果的归一化运算.从图 7 可以看出,GHARA 在 5 个测试程序中表现最好,其平均数据包延迟比 DOR 减少 35.4%,比 Local 减少 16%,比 RCA 减少 18%,比 GCA 减少 14%.其中在 raytrace 中,GHARA 表现了相对最优情况,其平均数据包延迟比 GCA 减少 40%.我们还可以看到在大部分实际测试程序中,全局自适应算法 GCA 比区域自适应算法 RCA 并没有太多的性能提升,这是由 GCA 在其传播拥塞信息的途径的局限性以及计算路由的速度决定的,而 GHARA 在传播拥塞信息和计算路由方面的设计保证了其在大部分情况下都能表现出良好的性能.

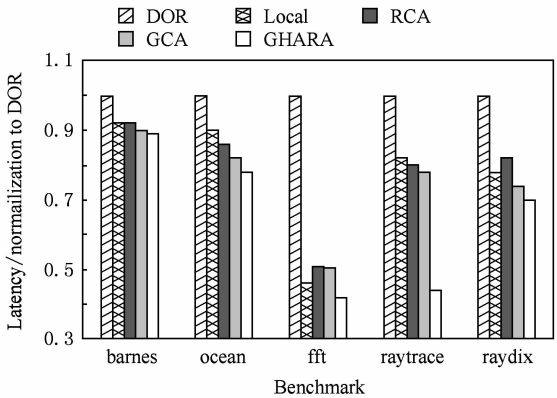
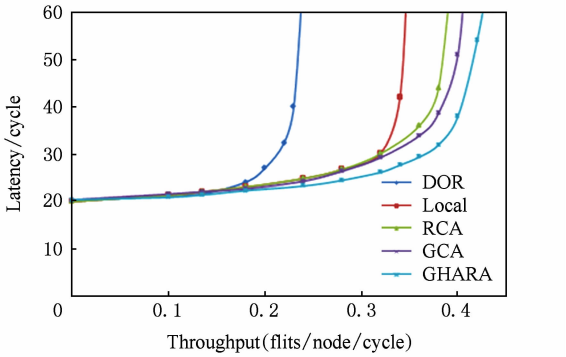


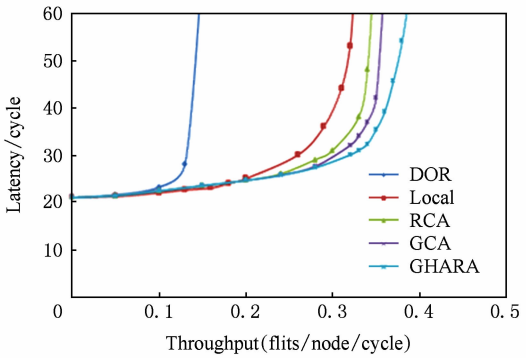
Fig. 7　Performance of SPLASH-2 benchmark traces.

图 7　SPLASH-2 测试程序集性能评估

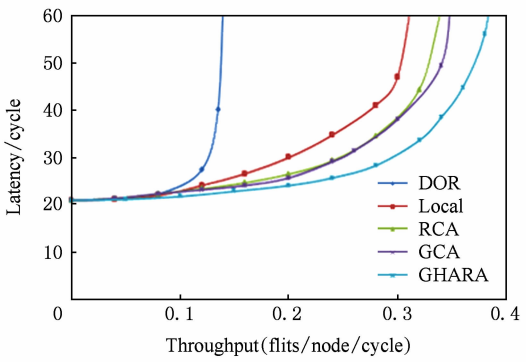
2) 人工合成通信.图 8 和图 9 分别给出了包括 GHARA 在内的 5 种路由算法在 8×8 和 16×16 的 2D Mesh 中不同负载下的平均数据包延迟,以此来衡量算法的性能.当延迟到达零负载延迟的 3 倍时,判断性能到达饱和点.可以看出在 3 种通信传输模式下,GHARA 都表现出了最好的性能.在刷洗(shuffle)、转置(transpose)和位反(bit-reverse)模式下,确定路由算法 DOR 都由于容易造成负载不均衡而严重影响了性能.自适应路由算法均在这 3 个模式中发挥了很好的作用,但是本地自适应算法 Local 以及区域自适应算法 RCA 均存在不同程度的近视现象,从而限制了算法的性能.而全局自适应路由算法 GCA 由于计算步骤较多,表现并不突出,8×8 Mesh 下平均饱和带宽仅比 RCA 提高 8.5%,在 16×16 Mesh 下的平均饱和带宽比 RCA 提高不足 6%.



(a) Shuffle



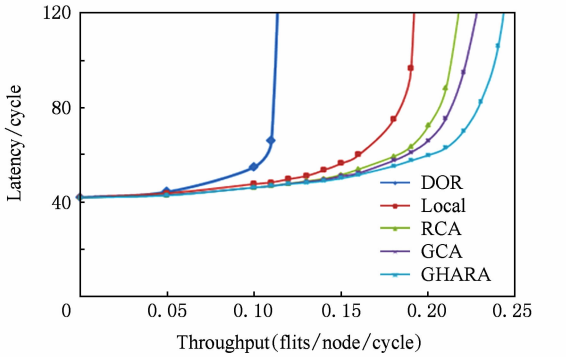
(b) Transpose



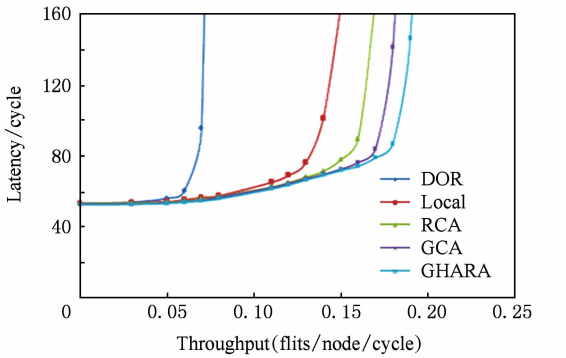
(c) Bit-reverse

Fig. 8 Performance of synthetic workloads (8×8 Mesh).

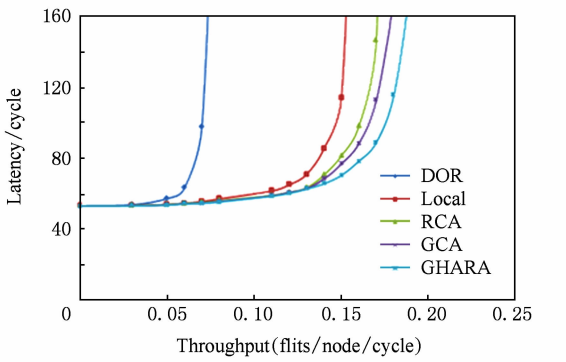
图 8 人工模式性能评估(8×8 Mesh)



(a) Shuffle



(b) Transpose



(c) Bit-reverse

Fig. 9 Performance of synthetic workloads (16×16 Mesh).

图 9 人工模式性能评估(16×16 Mesh)

GHARA 算法在 3 种模式下都胜过其他 4 种算法,表 1 和表 2 展示了实验中 8×8 和 16×16 Mesh 下所有算法在 3 种模式下的饱和带宽以及 GHARA 相对这些算法提高的百分比. 在 transpose 模式下,8×8 Mesh 下 GHARA 的饱和带宽比 GCA 提高 13.9%,16×16 Mesh 下 GHARA 的饱和带宽比 GCA 提高 17.6%. 3 个模式平均来看,GHARA 在 8×8 Mesh 下平均饱和带宽比 GCA 提高 10.7%,在 16×16 Mesh 下平均饱和带宽比 GCA 提高 14.7%. 随着网络规模的增加,本地自适应算法 Local 和区域自适应算法 RCA 由于近视现象产生次优路由越发明显,计算步骤较多的全局自适应路由算法 GCA 也越来越慢. 由于拥有全局拥塞信息同时计算速度较快,相比于其他算法 GHARA 在 16×16 Mesh 中比在 8×8 Mesh 中表现了更大的饱和带宽优势.

Table 1 Saturation Throughput on 8×8 Mesh and Improvement of GHARA

表 1 8×8 Mesh 饱和带宽及 GHARA 提高百分比

Algorithm	Shuffle	Transpose	Bit-reverse	Average Improvement of GHARA
DOR	0.24	0.15	0.14	1.37
Local	0.35	0.33	0.32	0.25
RCA	0.38	0.34	0.34	0.18
GCA	0.41	0.36	0.36	0.107
GHARA	0.44	0.41	0.41	

Table 2　Saturation Throughput on 16×16 Mesh and Improvement of GHARA

表 2　16×16 Mesh 饱和带宽及 GHARA 提高百分比

Algorithm	Shuffle	Transpose	Bit-reverse	Average Improvement of GHARA
DOR	0.11	0.07	0.07	1.56
Local	0.19	0.14	0.15	0.33
RCA	0.21	0.16	0.16	0.21
GCA	0.22	0.17	0.17	0.147
GHARA	0.25	0.20	0.19	

5　总结与展望

本文针对本地自适应算法和区域自适应算法很难有效获得全局负载信息以及目前全局自适应算法计算路由速度慢的问题,提出了一个新型全局自适应路由机制.该机制包含一个全局拥塞信息监视网络 Roof-Mesh 和一个全局自适应路由算法 GHARA. Roof-Mesh 将全网 Mesh 结构分成多级区域,针对不同区域中各个节点,得到不同精确度的全局节点拥塞信息.GHARA 是一个低开销、全局性、可扩展的多级自适应路由算法,其主要有 2 个特点:

1) 具有全局的视角.利用 Roof-Mesh 全局拥塞信息监视网络,每个节点获得不同精确度的全网各个节点的拥塞信息以指导路由计算,从而避免了本地自适应算法的短视现象.

2) 在已知各个节点拥塞度的条件下,逐级将 Mesh 全网络分解为 2×2 的区域计算路由.对比全局路由算法 GCA,计算步骤由从 $2\sqrt{N}-3$ 减少至 $\mathrm{lb}\,N$ 次.

实验结果表明:GHARA 表现优于其他区域和全局自适应路由算法如 RCA 和 GCA.在人工注入通信模式下 8×8 Mesh 平均饱和带宽比 GCA 提高 10.7%,16×16 Mesh 平均饱和带宽比 GCA 提高 14.7%.而在运行真实测试程序集 SPLASH-2 模式下,平均数据包延迟最多比 GCA 减少 40%,平均减少 14%.

GHARA 的可扩展性体现在随着网络规模的增加,我们只需要在全局拥塞信息监视网络 Roof-Mesh 中增加相应的拥塞监控单元、增大拥塞寄存器以及增加层次.我们接下来的工作要进一步优化全局拥塞信息监视网络 Roof-Mesh 的结构,提高处理速度;同时研究在其他拓扑结构上部署类似的结构,并在此基础上应用全局自适应路由算法.

参　考　文　献

[1] Wang Yongqing, Xie Lunguo, Fu Qingchao. Moveable bublle flow control and adaptive routing mechanism in torus networks [J]. Journal of Computer Research and Development, 2014, 51(8): 1854-1862 (in Chinese)
(王永庆, 谢伦国, 付清朝. Torus 网络中移动气泡流控及其自适应路由实现[J]. 计算机研究与发展, 2014, 51(8): 1854-1862)

[2] Sankaralingam K, Nagarajan R, Gratz P, et al. The distributed microarchitecture of the TRIPS prototype processor [C] //Proc of the 39th Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2006: 480-491

[3] Vangal S, Howard J, Ruhl G, et al. An 80-Tile1.28 TFLOPS network-on-chip in 65nm CMOS [C] //Proc of IEEE Int Solid-state Circuits Conf. Piscataway, NJ: IEEE, 2007: 98-99

[4] Rahman M, Shah A, Inoguchi Y. A deadlock-free dimension order routing for hierarchical 3D-mesh network [C] //Proc of Int Conf on Computer & Information Science (ICCIS). Piscataway, NJ: IEEE, 2012: 563-568

[5] Ramakrishna M, Gratz P, Sprintson A. GCA: Global congestion awareness for load balance in networks-on-chip [C] //Proc of the 7th Int Symp on Networks on Chip (NoCS). Piscataway, NJ: IEEE, 2013: 1-8

[6] Woo S, Ohara M, Torrie E, et al. The SPLASH-2 programs: Characterization and methodological considerations [C] //Proc of the 22nd Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 1995:24-36

[7] Dally W, Aoki H. Deadlock-free adaptive routing in multicomputer networks using virtual channels [J]. IEEE Trans on Parallel and Distributed Systems, 1993, 4(4): 466-475

[8] Li M, Zeng Q, Jone W. DyXY—A proximity congestion-aware deadlock-free dynamic routing method for network on chip [C] //Proc of the 43rd Design Automation Conf. Piscataway, NJ: IEEE, 2006: 849-852

[9] Gratz P, Grot B, Keckler S. Regional congestion awareness for load balance in networks-on-chip [C] //Proc of the 14th High Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2008: 203-214

[10] Ma Sheng, Jerger N, Wang Zhiying. DBAR: An efficient routing algorithm to support multiple concurrent applications in networks-on-chip [C] //Proc of the 38th Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2011: 413-424

[11] Manevich R, Cidon I, Kolodny A, et al. A cost effective centralized adaptive routing for networks-on-chip [C] //Proc of the 14th Euromicro Conf on Digital System Design (DSD). Piscataway, NJ: IEEE, 2011: 39-46

[12] Kim J, Park D, Theocharides T, et al. A low latency router supporting adaptivity for on-chip interconnects [C] //Proc of the 42nd Design Automation Conf (DAC). Piscataway, NJ: IEEE, 2005: 559-564

[13] Galles M. Spider: A high-speed network interconnect [J]. Micro, 1997, 17(1): 34-39

[14] Gratz P, Sankaralingam K, Hanson H, et al. Implementation and evaluation of a dynamically routed processor operand network [C] //Proc of the 1st Int Symp on Networks-on-Chip. Piscataway, NJ: IEEE, 2007: 7-17

[15] Kumar A, Kundu P, Singh A, et al. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS [C] //Proc of the 25th Int Conf on Computer Design (ICCD). Piscataway, NJ: IEEE, 2007: 63-70

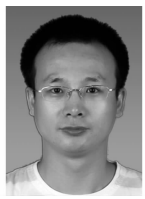
[16] Dally W, Towles B. Principles and Practices of Interconnection Networks[M]. San Francisco, CA: Morgan Kaufmann, 2003

[17] Binkert N, Beckmann B, Black G, et al. The GEM5 simulator [J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7

Zhang Yang, born in 1981. PhD candidate. Member of China Computer Federation. His main research interests include network-on-chips design and real-time scheduling.

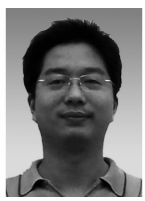
Wang Da, born in 1981. PhD, associate professor. Member of China Computer Federation. Her main research interests include IC testing and analysis, micro architecture design, many-core processor, and VLSI design and testing.

Ye Xiaochun, born in 1981. PhD, associate professor. Member of China Computer Federation. His main research interests include high-performance computer architecture, software simulation, algorithm paralleling and optimizing.



Zhu Yatao, born in 1978. PhD candidate. Member of China Computer Federation. His main research interests include high throughput computing architecture and software simulation.

Fan Dongrui, born in 1979. PhD, professor and PhD supervisor. Member of China Computer Federation. His main research interests include many-core processor design, high throughput processor design and low power micro-architecture.



Li Hongliang, born in 1975. PhD, associate professor. Member of China Computer Federation. His main research interests include high performance computing and processors architecture design.

Xie Xianghui, born in 1958. PhD, professor. Senior member of China Computer Federation. His main research interests include high performance computer architecture and distributed computing.