ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.9, September 2007, pp.2194−2204
DOI: 10.1360/jos182194

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

Xmesh:一个mesh-like片上网络拓扑结构*

朱晓静[1,2+], 胡伟武[2], 马 可[1,2], 章隆兵[2]

[1](中国科学技术大学 计算机科学技术系,安徽 合肥 230027)

[2](中国科学院 计算技术研究所 系统结构重点实验室,北京 100080)

Xmesh: A Mesh-Like Topology for Network on Chip

ZHU Xiao-Jing[1,2+], HU Wei-Wu[2], MA Ke[1,2], ZHANG Long-Bing[2]

[1](Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

[2](Key Laboratory of Computer Architecture, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62600853, Fax: +86-10-62527995, E-mail: zhuxiaoj@ict.ac.cn

Zhu XJ, Hu WW, Ma K, Zhang LB. Xmesh: A mesh-like topology for network on chip. *Journal of Software*, 2007,18(9):2194−2204. http://www.jos.org.cn/1000-9825/18/2194.htm

Abstract: Network on chip (NoC) has many characteristics, such as less nodes number, shorter distance between the cores, the need of less physical implementation difficulty, and so on. To satisfy the special need of the NoC, this paper presents a new topology named Xmesh and its routing algorithm called XM. This paper adds some diagonal edges on the Mesh topology to form Xmesh, sequentially reduce the average distances of the topology. Given the same network size, Xmesh has the same edge number with Torus topology, this paper compares the performance of Mesh, Xmesh and Torus topologies. A detailed theoretical analysis for Mesh, Xmesh and Torus topologies is given, and a simulation analysis based on the Popnet simulator using uniform traffic pattern and hotspot traffic pattern as benchmarks is also presented. As the simulation result shows, the average latencies of Xmesh and Torus topologies are less than 70% of the average latency of Mesh topology. For uniform traffic pattern, when the network size is small, the performance of Xmesh is better than Torus topology. For hotspot traffic pattern, when the hot node is near to the network center or the two diagonals, the latency of Xmesh is about 70%~90% of the latency of Torus topology, otherwise, the latency of Torus is about 70%~90% of the latency of Xmesh topology. In conclusion, Xmesh has a good performance just like Torus, but its physical implementation is simpler than Torus's, and both of which have a better performance than Mesh topology.

Key words: topology; routing algorithm; performance analysis; traffic pattern; network on chip

* Supported by the National Natural Foundation of China for Distinguished Young Scholars under Grant No.60325205 (国家杰出青年基金); the National Natural Science Foundation of China under Grant No.60673146 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2005AA110010, 2005AA119020 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2005CB321600 (国家重点基础研究发展计划(973)); the Natural Science Foundation of Beijing of China under Grant No.4072024 (北京市自然科学基金); Knowledge Innovation Program of the Institute of Computing Technology, the Chinese Academy of Sciences under Grant Nos.20056240, 20066012 (中国科学院计算技术研究所知识创新课题)

Received 2006-10-20; Accepted 2006-11-30

摘 要: 针对片上网络(network on chip,简称 NoC)的节点数量少、距离近、物理实现复杂度受到限制的特点,提出了一种新的 Xmesh 拓扑结构,并为该结构提出了 XM 路由算法.该结构在经典的 mesh 结构的基础上添加了两个对角线型的回边,缩短了节点间的距离,而且路由计算的复杂性不高,实现的复杂度基本没有增加.将 Xmesh 与经典的 Mesh 和 Torus 结构进行了理论分析比较,同时,在 Popnet 模拟器上基于均衡负载和热点负载两种负载模式进行性能比较.模拟结果表明,Xmesh 平均延时不到 Mesh 结构的 70%.对于均衡负载,当网络规模较小时,Xmesh 的延时比 Torus 的更小;对于热点负载,当热点距离网络中心或者对角线比较近时,Xmesh 的延时比 Torus 的小 10%~30%.反之,其延时比 Torus 的大 10%~30%.总的来说,Xmesh 的性能与 Torus 比较接近,但其物理实现更为简单,Xmesh 比 Mesh 结构的性能更好.

关键词: 拓扑;路由算法;性能分析;负载模式;片上网络

中图法分类号: TP393 **文献标识码**: A

随着芯片集成度的增加,未来单个芯片上的晶体管数目将达到数十亿个,片上多处理器能够有效利用片上的海量晶体管资源,这已成为当前高性能处理器的发展趋势[1−3].片上网络对片上多处理器的可扩展性及性能具有较为重要的影响,逐渐成为当前的研究热点[4−6].近年来,片上网络的研究主要集中于低功耗设计[6−8]、路由器设计[9,10]、实现与测试方法[11,12]和容错机制等.目前,关于拓扑结构和路由算法的研究不多,现有的单片多处理器大多采用经典的拓扑结构(包括Mesh,Torus等).由于片上网络的规模通常比大规模并行机的规模要小,在设计片上网络时又要充分考虑物理实现难度,所以,传统的用于并行机网络的Mesh,Torus等经典拓扑结构并不一定适用.因而针对其特点,研究适合片上网络的拓扑结构是很有意义的.

Carnegie Mellon大学的Ogras提出一种优化片上网络性能的方法[5],通过在Mesh上增加一些长距离的边以缩短节点间距离,减少路由延时.该方法虽然能够提高性能,但需要设计比较复杂的路由算法,实现难度大.Das提出新的Multi-Mesh(MM)拓扑结构[13],采用$n\times n$的Mesh结构作为拓扑结构的基本块.MM的理论延时比同等规模的Mesh要低一个数量级,但是对于小规模网络(如 16 个节点),其优势并不明显,而物理实现难度明显要高于Mesh.

我们目前正在进行龙芯三号的片上结构设计,设计目标是实现片上快速通信的单芯片多处理器.为了满足设计需求,针对片上网络的节点数量少、距离近、算法复杂度与物理实现难度不能很高的特点,本文提出一种新的 Xmesh 拓扑结构及相应的路由算法 XM,该拓扑结构适合各种网络规模,物理实现也较为简单.Xmesh 在 Mesh 结构上增加一些对称边,结构规则,路由算法简单.XM 是一种确定性非最短路径路由算法,它可以确保任意两个节点的路由路径长度不超过 Mesh 中的两点间距离.在小规模片上网络中,Xmesh 的性能优于 Mesh 和 Torus 两种经典拓扑结构.对于 $n\times n$ 的 Xmesh,任意两个节点间的距离不会超过 $n-1$,而在 Mesh 网络中,任意两个节点间距离的最大值是 $2\times(n-1)$;在 Torus 中,任意两个节点间距离的最大值是 n(n 为偶数时)或 $n-1$(n 为奇数时).与 Mesh 相比,Xmesh 大幅度缩短了节点间的距离;与 Torus 相比,Xmesh 的物理实现更为简单,这主要是因为 Xmesh 的回边比 Torus 要少(Xmesh 有 2 条,Torus 有 $2n$ 条).

本文第 1 节是相关工作.第 2 节描述 Xmesh 结构.第 3 节给出 Xmesh 的路由算法 XM.第 4 节从延时、路径多样性、吞吐量、网络直径 4 个方面对 Mesh,Torus,Xmesh 三种结构进行理论评估.第 5 节评测 XY,TXY,XM 这 3 种路由算法的性能.最后是总结.

1 相关工作

近年来,关于拓扑结构和路由算法的研究工作有:文献[14]中为Torus和Mesh结构提供了一种新的确定性路由算法,其核心思想是把原来的网络结构划分成若干个子网络,再把子网络分成相互独立的段;文献[15]中给出了三维六边形网络的建模方法、寻址方式以及路由算法.文献[6]中为fat tree拓扑结构提供了一种新的路由算法,目的是降低功耗和路由延时.文献[16]在片上网络中对 3 种拓扑结构进行性能分析,模拟并分析了ring,spidergon,Mesh这 3 种结构,结果显示,spidergon结构在功耗、面积、延时、可扩展性中可以得到很好的权衡.

下面,重点介绍Mesh,和Torus[17].

Mesh 结构:假定节点 X 的坐标是(i,j),$0\le i<n$,$0\le j<n$,任何一个节点 $Y(s,t)$只要满足$|s-i|+|t-j|=1$ 都与节点 X 相连.

为了给出具体的算法定义,首先做一些假定,设每次路由时当前节点坐标为 $C(cx,cy)$,目标节点为 $D(dx,dy)$,$xoffset=dx-cx$,$yoffset=dy-cy$,网络的规模是 $n\times n$.

二维 Mesh 结构的路由算法比较多.在确定性算法中,比较简单而且常用的有 XY 路由算法.

设输出端口方向为 out_dir,$X+$,$X-$,$Y+$,$Y-$分别表示 X/Y 轴正负方向,$Home$ 表示本地输出方向,XY 算法的伪代码见表 1.

Torus 结构:假定节点 X 的坐标是(i,j),$0\le i<n$,$0\le j<n$,任何一个节点 $Y(s,t)$只要满足下列 5 个条件之一就与节点 X 相连.若满足(2),(3),(4),(5)任意条件之一,那么 X 与 Y 之间的边称为回边.

(1) $|s-i|+|t-j|=1$;

(2) $i=s$,$j=0$,$t=n-1$;

(3) $i=s$,$j=n-1$,$t=0$;

(4) $i=0$,$s=n-1$,$j=t$;

(5) $i=n-1$,$s=0$,$j=t$.

二维 Torus 结构的确定性算法有 TXY 算法.TXY 至少需要两个虚通道(virtual channel,简称 VC)来保证不出现死锁,它的伪代码见表 1.

Table 1 Pseudocodes of XY and TXY routing algorithms

表 1 XY 与 TXY 路由算法的伪代码

XY routing algorithm	TXY routing algorithm

If (*xoffset*!=0)
{
  if (*xoffset*>0) *out_dir=X+*;
  else *out_dir=X–*;
}
else if (*yoffset*>0) *out_dir=Y+*;
else if (*yoffset*<0) *out_dir=Y–*;
else *out_dir=Home*;
	If (*abs*(*xoffset*)>*N*/2){
	  If (*xoffset*>0) {*out_dir=X–*;*vc*=1;} else {*out_dir=X+*;*vc*=0;}
	} else if (*xoffset*!=0){
	  if (*xoffset*>0) {*out_dir=X+*;*vc*=0;} else {*out_dir=X–*;*vc*=1;}
	} else if (*abs*(*yoffset*)>*N*/2){
	  if (*yoffset*>0) {*out_dir=Y–*;*vc*=1;} else {*out_dir=Y+*;*vc*=0;}
	} else if (*yoffset*!=0){
	  if (*yoffset*>0) {*out_dir=Y+*;*vc*=0;} else {*out_dir=Y–*;*vc*=1;}
	} else *out_dir=Home*;

2 Xmesh 拓扑结构

定义(Xmesh). 在 Mesh 结构上增加两条环形路径便形成 Xmesh.其中一条对角线上,对于任意 $0\le i<n-1$,节点$(i,n-1-i)$总与节点$(i+1,n-2-i)$相连,节点$(0,n-1)$也与节点$(n-1,0)$相连(通过回边);在另外一条对角线上,对于任意 $0\le i<n-1$,节点(i,i)总与节点$(i+1,i+1)$相连,节点$(0,0)$也与节点$(n-1,n-1)$相连(同样通过回边).

如图 1 所示,Xmesh 结构如同在原来的 Mesh 结构上增加了两个 X,其中一个 X 是由两个主对角线构成,另一个 X 由两条回边构成,所以命名为 Xmesh.$n\times n$ 的 Xmesh 比 $n\times n$ 的 Mesh 多 $2n$ 条边.

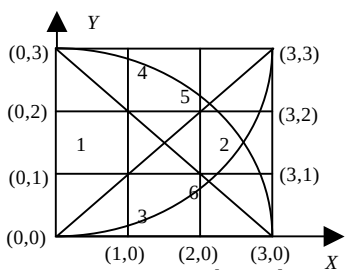


Fig.1 4×4 Xmesh topology

图 1 4×4 的 Xmesh 拓扑结构

3 Xmesh 的路由算法 XM

XM 是一种确定性非最短路径路由算法.该算法所得的路径未必是 Xmesh 结构的最短路径,但是这种路由算法可以保证所采用的路径长度不大于 Mesh 结构中的最短路径长度.在 Xmesh 上,我们也研究过最短路径路由算法和基于 Xmesh 的完全自适应路由算法,发现最短路径路由算法很容易造成两条主对角线的拥塞,因为大部分节点对间的最短路径都要经过主对角线,而完全自适应路由算法由于每次路由时有太多的选择,反而体现不出 Xmesh 结构本身的优势,因此,这两种算法我们最后都没有考虑,这里也不再对它们进行说明和评估(如图 2 所示).

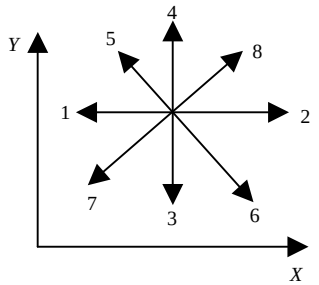


Fig.2 Eight routing directions of switch

图 2 路由 switch 的 8 个端口方向

XM 算法的伪代码如下:.

```
If (xoffset==0||yoffset==0) XY_algorithm(C,D);
else if (C 在主对角线的左边或右边){
    if (abs(xoffset)>=(n/2) and abs(yoffset)>=(n/2)){
        if (yoffset<0)  向上走;
        else  向下走;
    } else XY_algorithm(C,D);
} else if (C 在主对角线的上边或下边){
    if (abs(xoffset)>=(n/2) and abs(yoffset)>=(n/2)){
        if (xoffset<0)  向右走;
        else  向左走;
    } else YX_algorithm(C,D);
} else if (cx==cy){    //当前节点在主对角线上
    if (abs(xoffset)>=(n/2) and abs(yoffset)>=(n/2)){    //需要走回边
        if (xoffset<0)  沿对角线向右上方;
        else  沿对角线向左下方;
    } else {    //不需要走回边
        if (xoffset<0 and yoffset<0)  沿主对角线向下;
        else if (xoffset>0 and yoffset>0)  沿主对角线向上;
        else XY_algorithm(C,D);
    }
} else if (cx+cy==n−1){    //当前节点在反向主对角线上
    if (abs(xoffset)>=(n/2) and abs(yoffset)>=(n/2)){    //需要走回边
        if (xoffset<0)  沿反向对角线向右下方;
        else  沿反向对角线向左上方;
    } else {    //不需要走回边
        if (xoffset<0 && yoffset>0)  沿反向对角线向左上方;
        else if (xoffset>0 && yoffset<0)  沿反向对角线向右下方;
        else XY_algorithm(C,D);
    }
```

}

具体描述为:

设当前节点为 $C(cx,cy)$,目标节点为 $D(dx,dy)$;

1) 首先计算 $xoffset=dx-cx$,$yoffset=dy-cy$;若 $xoffset==0$ 或者 $yoffset==0$,则直接使用 XY 路由算法;否则,转 2).

2) 若当前节点 C 在两条对角线的左边或右边.首先判断条件 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$ 是否满足,若满足,说明目标节点离当前节点足够远,那么走回边比较近:进一步判断,若 $yoffset>0$,说明目标节点在当前节点上方,应该向下走,若 $yoffset<0$,说明目标节点在当前节点的下方,那么就应该向上走,这样反其道行之是为了充分利用回边;若不满足 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$,则不需要走回边,采用 XY 路由算法,即 X 优先,目的是尽可能地走对角线;否则,转 3).

3) 若当前节点 C 在两条对角线的上边或下边.首先判断条件 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$ 是否满足,若满足,说明目标节点离当前节点足够远,那么走回边比较近:进一步判断,若 $xoffset<0$,说明目标节点在当前节点的左边,应该向右走,若 $xoffset>0$,说明目标节点在当前节点的左边,应该向左走,这样的走反方向的目的也是为了充分利用回边;若不满足 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$,采用 YX 路由算法,即 Y 方向优先,这样做同样是为了尽可能地走对角线;否则,转 4).

4) 若当前节点在主对角线上,即满足 $cx==cy$ 的那条斜线.首先判断条件 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$ 是否满足,若满足,则说明目标节点离当前节点足够远,那么走回边比较近:此时,若 $xoffset<0$,说明目标节点在当前节点的左下方,应该走右上方的那条斜边,若 $xoffset>0$,说明目标节点在当前节点的右上方,应该走左下方的那条斜边,走反方向的目的也是为了充分利用回边;若不满足条件 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$,就要根据 $xoffset$,$yoffset$ 来判定怎么走:若 $xoffset>0$ and $yoffset>0$,则从当前节点的输出端口 8 出去,即走右上角;若 $xoffset<0$ and $yoffset<0$,则从输出端口 7 出去,即左下角;否则,采用 XY 路由算法.

5) 当前节点在反向主对角线上,即满足 $cx+cy==n-1$ 的那条斜线.首先判断条件 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$ 是否满足,若满足,则说明目标节点离当前节点足够远,那么走回边比较近:此时,若 $xoffset>0$,说明目标节点在当前节点的右下方,应该走左上方的那条边,若 $xoffset<0$,说明目标节点在当前节点的左上方,应该走右下方的那条边,走反方向的目的也是为了充分利用回边;若不满足条件 $abs(xoffset)>=n/2$ and $abs(yoffset)>=n/2$,进一步判断:若 $xoffset>0$,$yoffset<0$,则从当前节点的输出端口 6 出去,即右下角;若 $xoffset<0$,$yoffset>0$,则从当前节点的输出端口 5 出去,即左上角;否则,采用 XY 路由算法.

4 3 种拓扑结构的理论评估

Mesh,Torus 是目前应用最多的两种经典网络拓扑结构,本文将 Xmesh 结构与它们进行比较.下面从理想延时、路径多样性、理想吞吐量、网络直径、物理实现难度 5 个方面对 3 种拓扑结构 Xmesh,Mesh,Torus 进行评估.从得到的理论分析结果来看,从理想吞吐量、理想平均延时、网络直径 3 方面比较,Xmesh 都比 Mesh 结构的性能要好,Xmesh 的理想吞吐量、理想平均延时与 Torus 结构相同,Xmesh 的网络直径比 Torus 略小,它的路径多样性不如 Mesh 结构和 Torus 结构,Xmesh 的物理实现难度比 Torus 要小,但比 Mesh 要大.总的来说,Xmesh 与 Torus 结构各有各的优势,我们将从第 5 节可以看到,Xmesh 有一些特别的优点,是 Torus 结构所不具有的.

4.1 理想平均延时

理想平均延时[17]是指给定一个拓扑结构,假定在路由过程中完全没有拥塞,这个结构所有节点对之间的平均路由延时即为理想平均延时.文献[17]中给出了一个评估理想平均延时的公式:

$$T=H\times Tr+D/v+L/b \quad (1)$$

这里的 H 是指从源节点到目标节点的平均 hop 数,Tr 是在路由 switch 上的延时拍数,单位是 cycle/hop,D 是从源节点到目标节点的平均距离,通常情况下,H 等于 D,单位是 hop,v 是在线上的传输速度,单位是 hop/cycle,L 是包的长度,单位是 flit,b 是带宽,单位是 flit/cycle.在不同的拓扑结构中,H,D 的值是不一样的,Tr 则依赖于 switch 的物理实现.计算平均延时主要需要计算 H,也就是 D.文献[17]中给出了 Mesh 结构和 Torus 结构的节点间平均

距离公式.假定网络规模是 $n\times n$,如下所示:

$$H_{\min,T}=n/2 \tag{2}$$

$$H_{\min,M}=2n/3 \tag{3}$$

这个公式假定源节点和目标节点不会相同,但在我们的模拟数据中,源节点和目标节点是可以相同的,因此这个公式要略加改动,

$$H_{\min,T}=(n/2)\times(n^2-1)/n^2 \tag{4}$$

$$H_{\min,M}=(2n/3)\times(n^2-1)/n^2 \tag{5}$$

当 n=4 时,我们得到具体的数据

$$D_T=H_{\min,T}=1.875;D_M=H_{\min,M}=2.5.$$

Xmesh 结构是本文中新提出的拓扑结构,这里举例说明如何计算它的 D 值.在 4×4 的 Xmesh 上,同样可以把节点分成以上 3 类,节点间的距离见表 2.

Table 2 Distances from other nodes of three kind of nodes in 4×4 Xmesh

表 2 4×4 Xmesh 上的 3 类节点与其他节点的距离

	(0,0)	(0,1)	(1,1)
Nodes from which distance equals 1	(0,1),(1,0),(1,1),(3,3)	(0,0),(0,2),(1,1)	(0,0),(0,1),(1,0),(1,2),(2,1),(2,2)
Nodes from which distance equals 2	(0,2),(1,2),(2,2),(2,0),(2,1),(2,3),(3,2)	(0,3),(1,2),(2,1),(2,2),(3,3),(1,0)	(0,2),(0,3),(1,3),(2,3),(3,3),(3,2),(3,1),(3,0),(2,0)
Nodes from which distance equals 3	(0,3),(1,3),(3,0),(3,1)	(1,3),(2,3),(3,2),(3,1),(3,0),(2,0)	

因此,4×4 Xmesh 上的 3 类节点与其他节点间的距离之和分别为

$$D(0,0)=1\times4+2\times7+3\times4=30$$

$$D(0,1)=1\times3+2\times6+3\times6=33$$

$$D(1,1)=1\times6+2\times9=24$$

$$D_{xmesh}=(D(0,0)\times4+D(0,1)\times8+D(1,1)\times4)/(16\times16)=/256=1.875.$$

假定路由 switch 是 5 级流水,每两个流水级的间隔时间是 1 cycle,因此,在 switch 上处理的时间是 4 cycle,Tr=4 cycle/hop,线上传输延时 v=1 flit/cycle,L=2 flits,b=1 flit,因此,3 种结构的理想传输延时分别是

$$T_{mesh}=2.5\times4+2.5/1+2/1=14.5,$$

$$T_{torus}=1.875\times4+1.875/1+2/1=11.375,$$

$$T_{xmesh}=1.875\times4+1.875/1+2/1=11.375.$$

Torus 与 Xmesh 的理想平均延时相等,比 Mesh 结构的理想平均延时要小.

本文第 5 节的模拟数据可以验证这个理论计算结果的正确性.

4.2 路径多样性

路径多样性是指在一个拓扑结构中,若大多数节点对之间的最短路径个数大于 1,则称这个拓扑结构具有路径多样性[17].

通过简单的例子可以看到,Mesh,Torus 结构具有路径多样性,而 Xmesh 不具有路径多样性,Xmesh 中大多数节点对之间只有一条最短路径.如图 1 所示,在 4×4 的 Xmesh 中,节点对(0,0)和(3,3)之间只有一条最短路径,这两个节点间的距离为 1;而在 4×4 的 Mesh 中,节点(0,0)和(3,3)之间的最短路径数非常多,约为 19 条,这里的节点间距为 6.在 Torus 结构中,节点(0,0)和(3,3)之间的最短路径数为 2 条.可见,Xmesh 与 Mesh 结构相比,牺牲了路径多样性,换来了更短的节点间距离.与 Torus 结构相比,Xmesh 的节点间平均距离与 Torus 的一样长,但丢失了路径多样性,在这方面,Xmesh 稍微处于劣势.

4.3 理想吞吐量

理想吞吐量[17]是指对于给定拓扑结构,在完美的流控和路由机制下,网络中的最大吞吐量.文献[17]中给出了一个计算理想吞吐量的公式,其中,*Bc*是把整个网络分成均等的两半所需要的通道个数,*b*是每个通道的数据宽度,*N*是节点的总数:

$$TH \le 2b \times Bc/N.$$

对于 4×4 的 Mesh,*N*=16,*Bc*=8,要想把 Mesh 分成两半,需要切开 4 条边,每个边又都是双向的,所以 *Bc*=8;对于 4×4 的 Torus,*N*=16,*Bc*=16,要把 Torus 分成两半,需要切开 8 条边;对于 4×4 的 Xmesh,*N*=16,*Bc*=16,因为要切开的除了原属 Mesh 的 4 条边,还有新增的对角线上的两条边以及两条回边.

有了以上参数就可以得到 $THmesh \le b, THtorus \le 2b, THxmesh \le 2b$.总体来说,Xmesh 的理想吞吐量比 Mesh 的要大一些,与 Torus 的相同.

4.4 网络直径

网络直径(network diameter)是指网络中任意两节点间距离的最大值[18].通常,要提高网络传输的品质和速度,就需要减小网络直径.

在 $n \times n$ 的网络中,Mesh 结构的网络直径是 $2n-2$,Xmesh 的网络直径是 $n-1$(表 2 中已经举例说明).当 n 为奇数时,Torus 结构的网络直径是 $n-1$;当 n 为偶数时,Torus 结构的网络直径是 n.因此,Xmesh 结构的网络直径是最小的.

4.5 物理实现难度

物理实现难度主要取决于结构是否对称、结构的维数、回边的个数、路由 switch 端口数以及路由算法复杂度.Xmesh,Mesh,Torus 都是二维结构,Mesh 结构中所有互联边的长度都很短,并且 switch 的端口数都不大于 4,路由算法简单,因此物理实现难度最低;Xmesh 结构中大多数互联边的距离为 1,只有两条较长的回边,对角线上的 switch 端口数为 8,其余的 switch 端口数不超过 4,路由算法较为简单,因此,它的物理实现难度高于 Mesh 结构;Torus 结构中有较多的回边,路由 switch 端口数为 4,路由算法比较简单,因此,它的物理实现难度应该高于 Xmesh 结构.这里只是给出定性分析,具体量化比较有待进一步仿真验证.

5 性能分析

本节使用 3 种路由算法在 Mesh,Torus 和 Xmesh 这 3 种结构上进行模拟分析,这 3 种结构采用的算法分别是: Mesh 结构采用 XY 算法,Torus 结构采用 TXY 算法,Xmesh 结构采用 XM 算法.本文第 1 节和第 3 节已给出了相应的算法描述.

主要采用均衡负载和热点负载(hotspot)两种负载模式进行评估.这两种负载模式在时间上的分布都是完全均衡的,每隔一个常数时间都会生成一个包.均衡负载生成的包的源节点和目标节点都是随机数产生的;热点负载的特征是,每次生成一个包时,有 h 概率使目标节点是某一指定节点,其中概率 h 的值是可以调整的.

5.1 模拟环境

本文采用的Popnet模拟器由Princeton大学的Li Shang[19]开发而成,这是一个片上网络模拟器,可以很容易地嵌入到现有的多核模拟器中.Popnet采用wormhole流控机制,它的路由switch有 5 级流水,其基础拓扑结构是 *k*-ary *n*-cubes.模拟时设置的主要参数见表 3.其中,switch的 5 级流水如下:

a) *routing_decision*:判断包是否已经到达目的地,若否,则使用指定的路由算法为包选择若干可用通道;

b) *c_arbitration*:判断 output buffer 是否可用,从多个可用通道中随机选择一个通道;

c) *_arbitration*:在所有输出端口中并行执行这一操作,从所有端口请求中随机选择出一个,赋以输出端口的使用权;

d) *flit_outbuffer*:把选中的 FLIT 从 *input_buffer* 转移到 *output_buffer*,并给出上一 hop 的 switch CREDIT 信息;

e) *flit_traversal*:把 FLIT 放到输出线上,经过 WIRE_DELAY 的时间,下一 hop 的 switch 就可以接收到这个 FLIT.

若 FLIT 是包的头部,则会依次经历上面 5 个阶段;否则,判断是否到达目的地后,跳过 *vc_arbitration* 这个阶段,直接使用包头 FLIT 已得到的路由信息进行 switch 仲裁.

Table 3 The configuration of Popnet simulator

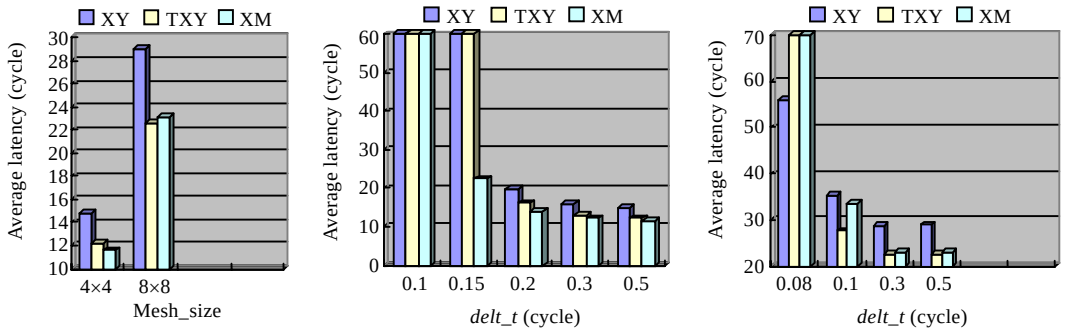
表 3 Popnet 模拟器的参数设置

Virtual channel number	4
Input buffer size	3flit
Output buffer size	3flit
Packet size	2flit

5.2 基于均衡负载的性能分析

如图 3(a)所示,当常数时间间隔 *delt_t*=0.5 时,对于均衡负载模式,XM 的平均延时比 XY 算法小很多,与 Torus 算法的平均延时基本持平.从图 3(a)可以看出,在负载强度比较低时,XM 比 XY 算法的延时约小 20%.同时还可以看出,在 4×4 的网络上,3 种拓扑结构的平均延时完全符合我们在第 4.1 节中计算出来的理论平均延时.

图 3(b)、图 3(c)分别用不同的网络 size 测试 3 种算法对于均衡负载的饱和吞吐量.总体来说,这 4 种算法达到负载饱和状态的条件差不多.在 4×4 的网络中,当负载的平均生成时间 *delt_t*=0.15 时,XY,TXY 算法的平均延时都大幅度增加,但 XM 在 *delt_t*=0.1 时才达到饱和状态.在 8×8 的网络中,3 种算法同时在 *delt_t*=0.08 时达到饱和.在达到饱和之前,4×4 的网络中 XM 延时最小,8×8 的网络中 TXY 的延时比 XM 的延时略小.



(a) *delt_t*=0.5 (b) 4×4 (c) 8×8

Fig.3 Performance analysis among three routing algorithms for uniform traffic pattern

图 3 3 种算法基于均衡负载的性能比较

从上面的模拟结果可知,对于均衡负载,当网络规模较大时,TXY 的性能比 XM 略好些;当网络规模比较小时,无论是平均延时还是饱和吞吐量,XM 的性能都是最好的,超过了 Torus 结构,远远优于 Mesh 结构,满足了我们设计这种拓扑结构的初衷.对于均衡负载,在较小规模的片上网络中,Xmesh 比 Mesh,Torus 两种经典拓扑结构更胜一筹.

5.3 基于热点负载的性能分析

首先,假定目标节点是热节点的概率为 0.3,即每次生成一个包时,有 30%的概率使目标节点为热节点,有 70%的概率随机生成目标节点.

5.3.1 中心热点的性能分析

这里首先选取的热节点是网络的中心节点,比如,对于 4×4 的网络,热节点就是(2,2);对于 8×8 的网络,热节点就是(4,4).

如图 4(a)所示,在 4×4 的网络上,以网络中心(2,2)为热节点.当时间间隔 *delt_t*=0.4 时,TXY 算法已经达到饱和,平均延时远远超出了另外两种算法.当时间间隔 *delt_t*=0.3 时,XY 达到饱和;当 *delt_t*=0.3,其他两种算法都已

经达到饱和时,XM 算法的延时依然很小.当 *delt_t*=0.5,3 种算法都还没有达到饱和时,XM 的平均延时最小.可见在 4×4 网络中,以网络中心为热点的情况下,无论负载强度高低,XM 算法的性能总是最好的.从图 4(b)中同样可以看出,在 8×8 的网络上,对于热节点在网络中心的热点负载,无论是平均延时还是饱和吞吐量,XM 算法都比 TXY 算法优异很多.当 *delt_t*=0.4 时,TXY 算法的延时已经有了显著上升.

由此可见,当热点处于网络中心时,无论网络规模大小,XM 的性能比其他两种算法都要好,此时,XY 和 TXY 两种算法的延时总是大于 XM 算法.

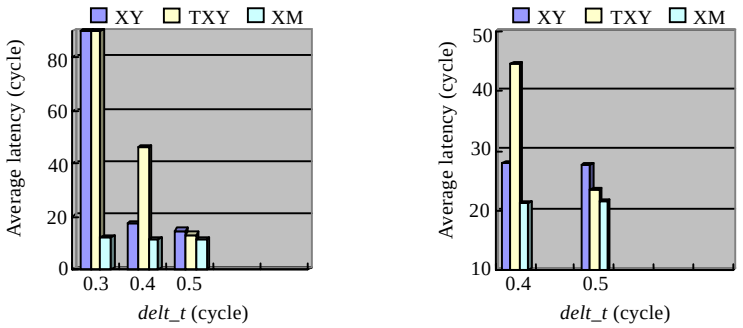


Fig.4 Performance analysis among three routing algorithms for hotspot traffic pattern

图 4 3 种算法关于热点负载的性能比较

5.3.2 其他热点的性能分析

不同位置的热节点具有不同的特性,这里从网络的不同位置选择热节点,而不仅仅局限在网络中心上.从图 6(a)可以看出,在 4×4 的网络中,即使负载强度不是很高,XY 算法的性能也比较差.而此时,XM 的性能很好,无论热节点处于哪种位置,XM 算法的性能在 3 种算法中都是最优的.

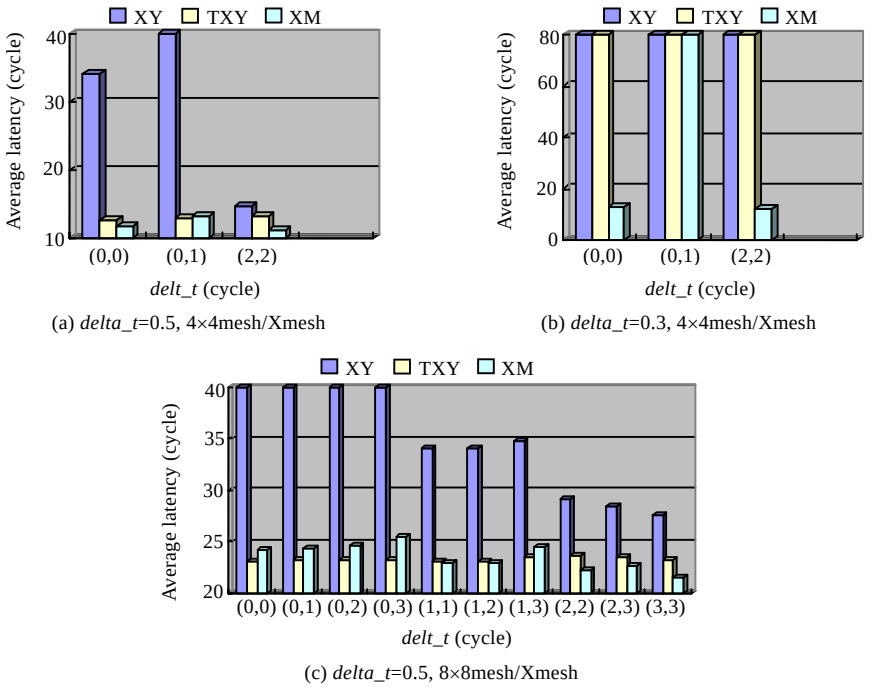
图 6(b)给出了当负载强度比较高时,4×4 网络上 3 种算法基于不同热节点的性能比较.此时,XY,TXY 两种算法都已经达到饱和,当(0,0),(1,1)分别作为热节点进行模拟时,XM 的延时与其理想延时相差不多.但是,当(0,1)作为热节点进行模拟时,XM 也达到了饱和.可见,当负载强度提高时,XM 在热点负载上的性能受热点位置影响很大.

由图 6(a)和图 6(b)可以得出结论,在 4×4 的网络规模中,无论热点处于什么位置、负载强度如何,Xmesh 结构的性能总是比其他两种结构的性能要好,再次满足了我们的设计目标,在小规模网络上达到最优.

图 6(c)给出了 8×8 的网络上负载强度较低时,3 种算法的性能比较.可以看出,在(0,3),(1,3)两个热点上,XM 的延时比 TXY 的延时要大一些,原因在于,此时热点与两个回边以及两条对角线的距离都比较远,很大一部分负载都用不上回边和对角线,XM 的优势发挥不出来.XY 算法对于热点负载的性能是相当差的.TXY 算法在中心热点上延时比 XM 的延时要大,但它的性能很稳定,因为在 Torus 结构中,所有节点都可以看作网络的中心.但正因为这种严格的对称性,在网络的中心节点上,TXY 算法的性能不如 XM,此时,XM 的节点间平均距离比较小,又能发挥它的对角线优势,因此可以得到更小的延时;当热点位于网络的 4 个顶点上,即(0,0),(0,*N*–1),(*N*–1,0),(*N*–1,*N*–1)时,XM 既有回边优势又有节点间平均距离比较远的劣势,优、劣势综合起来,XM 的延时与 TXY 基本相等.

因此,对于热点负载来说,网络上的节点分成 3 类:一种是对角线附近的节点,图 5(c)中 8×8 网络的 10 个热点中(0,0),(0,1),(1,1),(1,2)属于这类节点;一种是离网络中心较近或者在对角线上的节点,在图 5(c)给出的 10 个热点中,(2,2),(2,3),(3,3)属于这类节点;最后一种是处于离中心和对角线都较远的节点,图 5(c)的 10 个热点中,(0,2),(0,3),(1,3)属于这类节点.XM 在这 3 类节点上有着不同的性能.当第 1 种节点作为热点时,XM 与 TXY 算法性能基本相当,在路由过程中,如果源节点和目标节点相距很远,XM 可以发挥自己的回边优势;当第 2 种节点作为热点时,XM 的性能优于 TXY 算法,此时网络的平均距离比较小,又可以利用 XM 的两条对角线;当第 3 种节点作

为热点时,XM 的性能比 TXY 略差,因为这类节点离 Xmesh 结构的回边和对角线都很远.



(a) *delta_t*=0.5, 4×4mesh/Xmesh

(b) *delta_t*=0.3, 4×4mesh/Xmesh

(c) *delta_t*=0.5, 8×8mesh/Xmesh

Fig.5 Performance analysis while the hotpot node varies

图 5 热点位置变化时的性能分析

由上述模拟结果可知,在小规模网络上,Xmesh 结构的性能优于 Mesh,Torus 结构,当网络规模较大时,Xmesh 与 Torus 结构各有所长,Mesh 结构的性能最差.

对于均衡负载,XM 和 TXY 算法比 XY 的延时约小 20%;当网络规模较大时,TXY 的延时比 XM 的延时约小 5%;当网络规模较小时,XM 的延时比 TXY 约小 5%~20%,此时,XM 性能最佳,达到我们最初设计这种结构的目的和要求,在小规模网络上优于 Mesh 和 Torus 两种经典结构.

对于热点负载,当网络规模较小时,XM 在 3 种算法中性能最优,它的延时比 TXY 约小 10%~20%;当网络规模较大时,热点的位置对 XM 的性能影响很大.若热节点处于网络的中心,XM 的性能在 3 种算法中最好;当热点处于网络的对角线附近时,XM 的性能与 TXY 的性能相近;当热点离中心和对角线都较远时,XM 的性能比 TXY 的性能略差.XY 算法性能最差,非常不适用于热点负载.

Xmesh 结构的优点在于缩短了节点间的最短路径.我们提出的 XM 算法有一定的不足,它过于依赖对角线,当网络的负载增多时,会有很多的包拥塞在对角线上等待处理.因此,XM 算法更适用于小规模网络,还需要进一步改进.

6 结 论

Xmesh 是一种类似于 Mesh 的拓扑结构,在小规模片上网络,它的性能超过 Mesh 和 Torus 结构;对于较大规模的片上网络,Xmesh 的性能优于 Mesh 结构,某些情况下也优于 Torus 结构.我们目前只是在模拟器中实现并分析这个结构的性能,还需要进一步考虑 Xmesh 结构和算法的物理实现难度.由于 Xmesh 的最短路径通常都要经过两条主对角线,还需要进一步调整路由算法,增加路由算法自适应性,以免不必要的拥塞降低 Xmesh 结构的性能.

References:

[1] Kalla R, Sinharoy B, Tendler JM. IBM power5 chip: A dual-core multithreaded processor. IEEE Micro, 2004,24(2):40−47.

[2] Kevin K. Best servers of 2004: Where multicore is the norm. MicroProcessor Report, 2005.

[3] Kongetira P, Aingaran K, Olukotun K. Niagara: A 32-way multithreaded sparc processor. IEEE Micro, 2005,25(2):21−29.

[4] Bartic TA, Mignolet JY, Nollet V, Marescaux T, Verkest D, Vernalde S, Lauwereins R. Highly scalable network on chip for reconfigurable systems. In: Proc. of the Systems on Chip Conf. 2003. 79−82. http://ieeexplore.ieee.org/iel5/8954/28361/01267722.pdf?tp=&arnumber=1267722&isnumber=28361

[5] Ogras UY, Marculescu R. It's a small world after all, NoC performance optimization via long link insertion. IEEE Trans. on Very Large Scale Integrat. System. http://www.ece.cmu.edu/~sld/pubs/papers/TVLSI_July06_Special_issue.pdf

[6] Nickray M, Dehyadgari M, Afzali-Kusha A. Power and delay optimization for network on chip. Circuit Theory and Design. 2005,3: 273−276. http://ieeexplore.ieee.org/iel5/10211/32580/01523113.pdf?arnumber=1523113

[7] Simunic T, Boyd SP. Managing power consumption in networks on chips. IEEE Trans. on Very Large Scale Integration (Vlsi) Systems, 2004,12(1). http://www.stanford.edu/~boyd/reports/pwr_noc.pdf

[8] Banerjee N, Vellanki P, Chatha KS. A power and performance model for network-on-chip architectures. In: Proc. of the Design, Automation and Test in Europe Conf. and Exhibition Volume II (DATE 2004). 2004. http://portal.acm.org/citation.cfm?id=969210

[9] Kim J, Park D, Theocharides T, Das C, Vijaykrishnan N. A low latency router supporting adaptivity for on-chip interconnects. In: Proc. of the Design Automation Conf. 2005. 559−564. http://ieeexplore.ieee.org/iel5/10211/32580/01523113.pdf?arnumber=1523113

[10] Rijpkema E, Goossens KGW, Radulescu A, Dielissen J, van Meerbergen J, Wielage P, Waterlander E. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In: Proc. of the Design, Automation and Test in Europe Conf. and Exhibition (DATE 2003). 2003. http://citeseer.ist.psu.edu/564998.html

[11] Portero A, Pla R, Carrabina J. SystemC implementation of a NoC. Industrial Technology, 2005. 1132−1135.

[12] Stewart K, Tragoudas S. Interconnect testing for networks on chips. In: Proc. of the 24th IEEE VLSI Test Symp. (VTS 2006). 2006. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1617570

[13] Das D, De M, Sinha BP. A new network topology with multiple meshes. IEEE Trans. on Computers, 1999. 536−551.

[14] Mejia1 A, Flich1 J, Duato1 J, Reinemo S-A, Skeie T. Segment-Based routing: An efficient fault-tolerant routing algorithm for meshes and Tori. In: Proc. of the IEEE Int'l Parallel & Distributed Processing Symp. 2006. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1639341

[15] Decayeux C, Seme D. 3D hexagonal network: Modeling, topological properties, addressing scheme, and optimal routing algorithm. IEEE Trans. on Parallel and Distributed Systems, 2005. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1490517

[16] Bononi L, Concer N. Simulation and analysis of network on chip architectures: Ring, spidergon and 2D mesh. In: Proc. of the Design, Automation and Test in Europe. 2006. http://portal.acm.org/citation.cfm?doid=1131355.1131388

[17] Dally J, Towles B. Principles and Practices of Interconnection Network. Morgan Kaufmann Publishers, 2003.

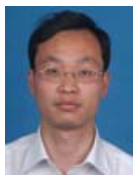
[18] Shen Z. Average diameter of network structures and its estimation. In: Proc. of the 1998 ACM Symp. on Applied Computing. 1998. 593−597. http://portal.acm.org/citation.cfm?id=330560.330973

[19] http://www.princeton.edu/~lshang/popnet.html

朱晓静(1982－),女,河南长垣人,博士生,主要研究领域为计算机系统结构,片上网络.

马可(1980－),男,博士生,主要研究领域为计算机系统结构.

胡伟武(1968－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为计算机系统结构.



章隆兵(1974－),男,博士,副研究员,主要研究领域为微处理器设计,并行处理.