一种递归定义的可扩展片上网络拓扑结构

朱晓静

(中国科学院计算技术研究所计算机系统结构重点实验室 北京 100190)
(北京龙芯中科技术服务中心有限公司 北京 100190)

摘 要 晶体管工艺的持续发展导致片上处理器数的逐渐增多,片上系统的核间通信要求吞吐量高、延时低、可扩展性好,传统的片上总线和 crossbar 互连结构已无法满足片上系统的通信需求,为此研究者提出新的片上互连结构,称为片上网络.为满足片上网络的特有通信需求,提出了一种可扩展的拓扑结构 Rgrid 及其路由算法 DR,它缩短了片上处理器间的平均距离并且比 Torus 结构容易实现.作者在龙芯用户级模拟器上分别实现 Mesh 和 Rgrid 结构,运行 Splash2 并行测试程序集比较这两种结构的性能优劣.运行程序结果表明,Splash2 程序在 Rgrid 结构中运行得到的 IPC 比在 Mesh 结构增加了 0.5%~148%,Rgrid 结构的平均延时比 Mesh 小 5%~81%.

关键词 拓扑结构;片上网络;可扩展;性能优化
中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2011.00924

A Recursive Scalable Topology for Network on Chip

ZHU Xiao-Jing

(*Key Laboratory of Computer System and Architecture*, *Chinese Academy of Sciences*, *Beijing* 100190)
(*Loongson Technology Corporation Limited*, *Beijing* 100190)

Abstract The development of integrated circuits makes the number of on-chip cores increase. Communication among the cores demands higher throughput, lower latency and more scalability. Traditional on-chip bus can not satisfy the need of on-chip communication. So researchers present a new interconnect architecture, called network on chip. In order to meet the special demand of network on chip, this paper gives a scalable topology named Rgrid and its routing algorithm called DR. Rgrid can reduce the average hops between on-chip cores, whose physical implementation is much easier than Torus topology. The author implements the Rgrid and Mesh tolopogies in the Godson3 simulator. The simulation results show that, simulator can gain much better performance using Rgrid topology than using Mesh topology for the Splash2 benchmarks. Compared to Mesh topology, the IPC of benchmarks of Rgrid increases by 0.5%~148%, the average latency degrades by 5%~81%.

Keywords topology; network on chip; scalable; performance optimization

1 引 言

由于晶体管工艺的发展和处理器主频的快速增长,片上总线的带宽和延时已满足不了片上多核通信的要求,总线的长连线延时成为片上系统性能进一步提高的瓶颈.为了更好地组织片上数目众多的处理器核,设计者需要一个模块化、扩展性好、可重用、高性能的互连结构.2001 年,Dally、Sgroi 等[1-2]分别提出了片上网络(Network on Chip)互连结构,简称

收稿日期:2007-01-04;最终修改稿收到日期:2008-07-14.本课题得到国家科技重大专项(2009ZX01028-002-003,2009ZX01029-001-003)和国家自然科学基金(60736012, 60921002,61050002,61003064,61070025)资助.朱晓静,女,1982 年生,博士,助理研究员,主要研究方向为计算机系统结构. E-mail: zhuxiaoj@ict.ac.cn.

NoC. 基于 Switch 结构的片上网络能克服片上总线吞吐量低、连线长的缺点，在片上网络中可以有多个核同时传输多条信息，提高了通信带宽，使得片上资源利用率更高. 片上网络技术可以提高系统的性能和可扩展性并降低片上系统的面积和功耗.

近年来片上网络的研究包括拓扑结构的性能分析、低功耗设计、路由器结构设计、容错算法设计、测试与实现方法等等. 片上网络的性能标准有吞吐量、传输延时、功耗、面积等. 在设计片上网络时，除了要选择拓扑结构、路由算法，还要考虑路由 switch 的结构，如端口的 FIFO、arbiter、MUX、虚通道等. 如何在设计时选择合适的结构和算法，需要大量的性能分析.

常用的网络拓扑结构有 2D-mesh、Torus、Ring、Hypercube、Star、Fat-tree 等. 这些结构以前主要应用于大规模并行机的互联系统，近年来研究者将这些结构在片上网络中实现并对其进行性能分析[3]. 片上系统与大规模并行系统的设计标准差异很大，设计时要考虑功耗、面积、硬件实现难度等问题. 由于片上连线更细，片上系统可以在同样面积内布置更多线，得到更高的带宽；片上系统要求通信延时跟时钟周期处于同一数量级，需要更低的通信延时，同时 switch 结构不能过于复杂，拓扑结构的设计要充分考虑布线的实现难度.

目前在经典拓扑结构基础上进行性能分析的研究较多，有研究者提出新的网络拓扑结构[4]，但很少有人专门为片上系统提出新的拓扑结构. Pavlidis 等提出在片上系统中使用 3-D 拓扑结构[5]，这样做无疑可以减小节点之间的跳数(hop)，但却给物理设计带来了极大挑战. 片上系统跟大规模并行系统的特征存在较大差异，设计一个适用于片上网络，能够提供低延时、高带宽、硬件实现难度不高的拓扑结构是很有意义的.

本文第 2 节介绍相关工作；第 3 节给出 Rgrid 拓扑结构的定义、结构特征及硬件实现难度；第 4 节定义 Rgrid 的路由算法 DR；第 5 节在龙芯用户级模拟器上分析 Splash2 程序在 Rgrid 与 Mesh 结构的性能优劣；最后是总结.

2　拓扑结构和路由算法

拓扑结构和路由算法的选择对于片上网络系统性能影响很大. Switch 间的联线共同组成片上网络的拓扑结构，拓扑结构的选择不仅影响延时与吞吐量，还会影响片上面积和功耗. 拓扑结构可以分为二维结构和多维结构，二维结构在片上实现更容易些，但多维结构的节点间距离(以 hop 计数)通常小一些.

在片上系统研究中最常用的拓扑结构是 Mesh 结构，这种结构通常采用 XY 路由算法，XY 是一种无死锁的最短路径路由算法，实现简单，路由的硬件开销较小，本文的性能分析部分 Mesh 结构就采用了这种算法. 研究结果表明[6]，同样的网络规模下，Mesh 结构的网络面积较大. Mesh 结构虽然实现简单，但是节点间平均距离(hop)比较大，因此直接导致 Mesh 结构的通信延时比较长.

Pande[3] 详细分析了几种拓扑结构的性能，这些结构包括 SPIN[7]、CLICHÉ[8]、Torus[2]、Folded torus、Octagon[9] 和 BFT[10]. 可扩展性是衡量片上系统性能的一个重要标准，Farahabady 提出了一种金字塔形的可扩展拓扑结构，称为 WK- Pyramid[4]，这是一个可以递归定义的三维拓扑结构，该结构的优点是网络直径和节点间平均距离较小从而路由速度比较快，但三维的拓扑结构和密集的节点间连线使得物理布局布线难度增大，因而难以在片上系统实现.

路由算法可以分为健忘性算法和自适应性算法两类，其中健忘性算法在路由过程中完全不考虑当前状况，只根据源节点和目标节点的位置确定所走的路径，自适应性算法又分为确定性和随机算法两类. Daeho Seo[11] 为 2D-mesh 提出了一种新的路由算法O1TURN，并将其与 DOR、VALIANT、ROMM 一系列健忘性算法作比较. 因此设计新的路由算法对于片上网络的性能优化有一定的帮助.

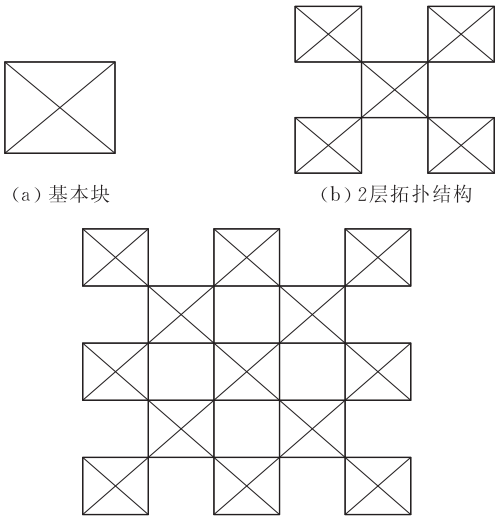
3　Rgrid 结构的定义、结构特征、硬件实现难度

3.1　Rgrid 结构定义

本文提出一种新的可扩展的网络拓扑结构，命名为 recursive-grid，简称 Rgrid，如图 1 所示.

定义 1(Rgrid).　Rgrid 的形式化定义为 $R_{i+1}=R_i \bigcup (4i \times R_1)$. 意思即为 $i+1$ 层结构 R_{i+1} 是由 i 层结构 R_i 和 $4i$ 个基本块组成的.

可见 Rgrid 是递归定义的，它的基本结构 R_1 如图 1(a)所示，由 4 个全相连的节点组成，这是拓扑结构定义的基本块，2 层结构 R_2 的定义如图 1(b)所示，分别在 R_1 的 4 个顶点上再添加 4 个基本块，3 层结构 R_3 的定义如图 1(c)所示，在 R_2 的所有外部顶点上共添加 8 个基本块.



(a) 基本块 (b) 2层拓扑结构

(c) 3层拓扑结构

图 1

3.2 Rgrid 结构特征

节点数 $N(n)$. 基本结构的节点数为 $4=(2\times1)^2$, 2 层结构的节点数为 $16=(2\times2)^2$, 3 层结构的节点数为 $36=(2\times3)^2$. 因此可以归纳得出结论:对于 n 层结构,节点总数 $N(n)=(2\times n)^2$.

边数 $E(n)$. 基本结构的边数为 6, 2 层结构的边数为 $6\times5=30$, 3 层结构的边数为 $6\times13=78$, 归纳可知,对于 n 层结构,边的总数为 $E(n)=(2^{n+1}-3)\times6$.

基本块数 $B(n)$. 基本结构的基本块的个数为 $1=2^2-3$, 2 层结构的基本块的个数为 $5=2^3-3$, 3 层结构的基本块的个数为 $13=2^4-3$,归纳可知,对于 n 层结构基本块的个数为 $B(n)=2^{n+1}-3$.

直径 $D(n)$. 基本块直径为 1,2 层结构的直径为 3, 3 层结构的直径为 5,归纳可知,n 层结构的直径 $D(n)=2n-1$.

节点间平均距离 $H(n)$. $H(n)$的定义是,$H(n)=\sum distance(i,j)/N^2$,意思即为节点间平均距离是所有的节点间距离之和除以节点对的个数,其中 i,j 表示节点,$distance(i,j)$表示节点 i 和 j 之间的距离,N 表示节点总数.

对于 1 层结构,所有节点间距离都是 1,因此 $H(1)=1$; 2 层结构中,共有两种不同位置的节点,$H(2)=((1\times6+2\times9)\times4+(1\times3+2\times3+3\times9)\times12)/(16\times16)=(96+36\times12)/256=528/256=2.06$; 3 层结构中,有 5 种不同位置的节点,$H(3)=((1\times3+2\times3+3\times9+4\times7+5\times13)\times12+(1\times6+2\times9+3\times7+4\times13)\times4+(1\times6+2\times11+3\times10+4\times8)\times8+(1\times3+2\times6+3\times14+4\times4+5\times8)\times8+(1\times6+2\times16+3\times13)\times4)/(36\times36)=(129\times12+97\times4+90\times8+113\times8+77\times4)/(36\times36)=2.98$.

综合上述结果,对于 $N\times N$ 的 Mesh,节点间的平均距离 $H_m=2N/3$[12],边数为 $E_m=2N^2-2N$,直径 $D_m=2N-1$;对于 $N\times N$ 的 Torus,若 N 是偶数,那么节点间平均距离 $H_t=N/2$[12],边数 $E_t=2N^2$,直径 $D_t=N$. 2 层 Rgrid 的节点数为 $(2\times2)^2$,与 4×4 的 Mesh/Torus 规模相等, 3 层 Rgrid 的节点数为 $(2\times3)^2$,与 6×6 的 Mesh/Torus 规模相等.

比较这 3 种拓扑结构的参数,当 $N=4$ 时,Mesh、Torus 与 2 层 Rgrid 的节点数同为 16,比较三者的平均距离:$H_m=8/3$, $H_t=2$, $Hr=2.06$;比较直径:$D_m=6$,$D_t=4$,$D_r=3$;比较边数: $E_m=24$, $E_t=32$, $E_r=30$. 当 $N=6$ 时,Mesh、Torus、3 层 Rgrid 的节点数同为 36,比较三者的平均距离:$H_m=4$,$H_t=3$,$Hr=2.98$;比较直径:$D_m=10$,$D_t=6$,$D_r=5$;比较边数: $E_m=60$,$E_t=72$,$E_r=78$. 由此可见,当网络规模相同时,Rgrid 的网络直径最小,节点间平均距离小于 Mesh 结构,略大于 Torus 结构,当网络规模小于等于 10×10, 3 种结构的边数基本属于同一数量级,仅当网络规模大于 5 层 Rgrid 时,Rgrid 的边数才会远远超出 Mesh 和 Torus 结构.

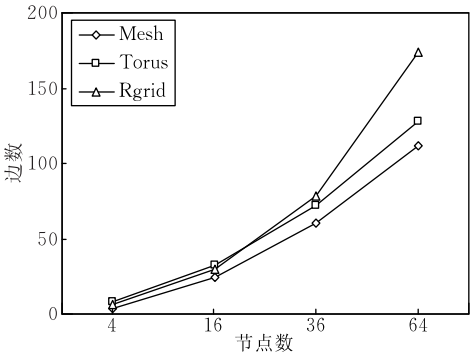
图 2 对 Mesh、Torus、Rgrid 3 种结构的理论特性作了直观比较. 综合来看,Rgrid 增加了结构中的连线个数从而缩短节点间距离. Rgrid 的节点间平均距离和直径跟 Torus 相近,都比 Mesh 结构小. 下面一小节给出说明,Rgrid 的硬件实现难度比 Torus 低,更容易实现.

3.3 Rgrid 硬件实现难度与开销分析

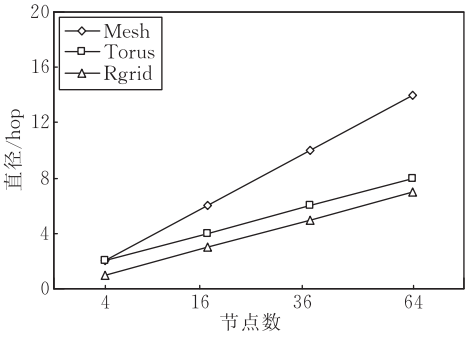
3.3.1 连线延时分析

对于片上网络,拓扑结构的物理布线直接影响处理器间的线延时,如果互连结构的线延时太长会使得在一拍内数据无法从一个节点传输到下一个节点. 文献[3]中给出图 3 所示几种拓扑结构,布线最方便的应该是 Mesh 结构,在 Mesh 结构中所有边的长度都相等,由于有较长的回边,Torus 结构的布线变得困难,因此在互连结构中通常使用 Torus 结构的另外一种形式:Folded-torus. Folded-torus 可以缩短最长的线长度,从而它的线延时比 Torus 结构小.

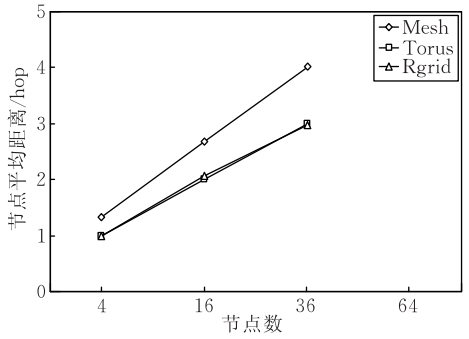
片上系统的物理布线通常都是水平和垂直布线,因此在 Rgrid 结构中,斜边要经过水平和垂直两个方向,它的长度是水平或垂直边的二倍,这样长度



(a) 节点相等时3种拓扑结构的边数比较



(b) 节点相等时3种拓扑结构的直径比较



(c) 节点相等时3种拓扑结构的节点平均距离比较

图 2　3 种结构的特性比较

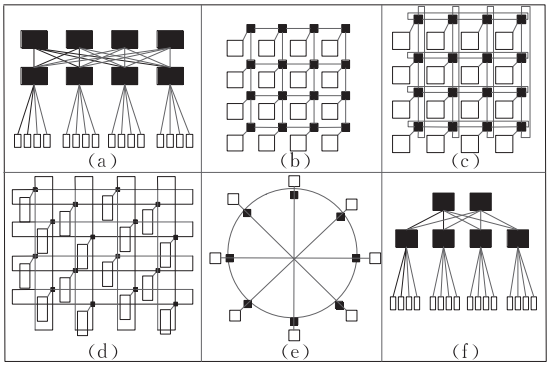


图 3　几种经典拓扑结构

的线延时可以在一拍之内完成，如图 1 所示，网络规模扩大不会引起 Rgrid 中的边增长，而 Torus 结构中最长边的长度跟网络节点数的平方根成正比，Rgrid 的扩展性很好. 因此 Rgrid 的硬件实现难度跟 Mesh 结构相近，比 Torus 结构容易实现.

3.3.2　面积和功耗分析

片上网络中，功耗面积开销较大的部分主要是 buffer 模块和路由器间连线. Alpha21364 的路由器和连线功耗共有 23W[13]，其中 58% 是由连线电路消耗的. Rgrid 增加了路由器间端口数，也增加了路由器间连线数目，从而增加了片上的功耗和面积，每个路由器的端口数目从原来的 5(本地端口加上 4 个外部方向)增加到 7(本地端口加上 6 个外部方向)，路由器间连线数大约增加了 40%，输入输出端口间的连线数目大约增加了一倍(连线数从 $5^2=25$ 增加到 $7^2=49$)，因此路由器本身的连线功耗和面积大约增加了一倍. 从这个方面进行分析，Rgrid 比 Mesh 结构增加了大量的片上功耗与面积开销，但增加的开销不会超出原来的一倍.

4　Rgrid 的路由算法 DR

DR(Deterministic routing algorithm of Rgrid)算法，即 Rgrid 的确定性算法. 其实现如图 4 所示，该算法不是最短路径算法，但所得路径长度最多是最短路径算法长度加 1，如图 3(b)所示，从源节点 $S(2,2)$ 到目标节点 $D(3,0)$，路径长度为最短路径长度加 1，而从源节点 $S(3,5)$ 到目标节点 $D(5,2)$ 走的路径则是最短路径.

如图 1 所示，Rgrid 结构的边有 4 个路由坐标轴，分别记为 X,Y,XY(沿着这个方向 $x++,y++$ 或者 $x--,y--$)，YX(沿着这个方向 $x++,y--$ 或者 $x--,y++$). 在 Rgrid 拓扑结构中，内部节点的度数都是 6，即每个节点有 6 个邻节点，X,Y,XY 或者 YX 三个方向各两条边. 设当前节点为 $C(cx,cy)$，目标节点为 $D(dx,dy)$，若 $(cx+cy)\%2==0$ 则表示当前节点有 XY 方向上的边，若 $(cx+cy)\%2==1$ 表示当前节点有 YX 方向的边.

有一点需要注意，外围节点跟内部节点不同，它们只有 3 条邻边，因此当前节点或者目标节点在外围时要格外注意有些边是不存在的. 举例说明，在 2 层 Rgrid 结构上，节点(1,0)和(2,0)不相邻，若要从节点(1,0)到(2,0)路由，就只能先走 $Y+$ 方向，路经(1,0)—>(1,1)—>(2,1)—>(2,0).

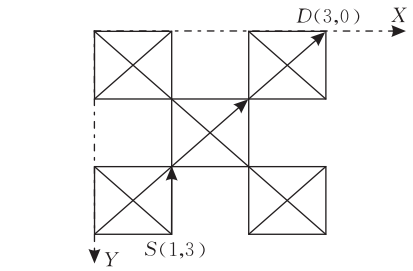
DR 算法包括两个函数. 其中 DR 函数先处理目标节点的边缘情况，然后调用 MIN 函数，赋给 MIN 函数的参数可以保证目标节点是内部节点，这样 MIN 函数只需要处理当前节点位于边缘的情况，然后作出路由选择. 这两个函数的伪代码如图 4 和图 5 所示. DR 算法的路由过程如图 6 所示.

```
MIN(cx,cy,dx,dy)
{//调用本函数之前,已经保证源节点跟目标节点绝对不相同,且
   目标节点是内部节点
 if ((cy==0‖cy==mesh_width-1) &&
   ((cx%2==1 && dx>cx)‖(cx%2==0&&dx<cx)){
   //若当前节点在最上面或最下面的水平线上,并且在 X 方向走
     不动,那么先走 Y 方向
   If(dy>cy) 走 Y+方向; else 走 Y-方向;
 }else if ((cx==0‖cx==mesh_width-1)&&
        (((cy%2==1&&dy>cy)‖(cy%2==1&&dy<cy)))){
   //若当前节点在最左边或者最右边的垂直线上,并且在 Y 方向
     走不动,那么先走 X 方向
   if(dx>cx) 走 X+方向; else 走 X-方向;
 }else if (cx<dx &&cy<dy &&(cx+cy)%2==0) 走XY+方向;
 else if (cx>dx && cy>dy &&(cx+cy)%2==0) 走XY-方向;
 else if (cx<dx && cy>dy &&(cx+cy)%2==1) 走YX+方向;
 else if (cx>dx && cy<dy &&(cx+cy)%2==1) 走YX-方向;
 else if (abs(cx-dx)>=abs(cy-dy)) 走 X 方向;
 else 走 Y 方向;
}
```

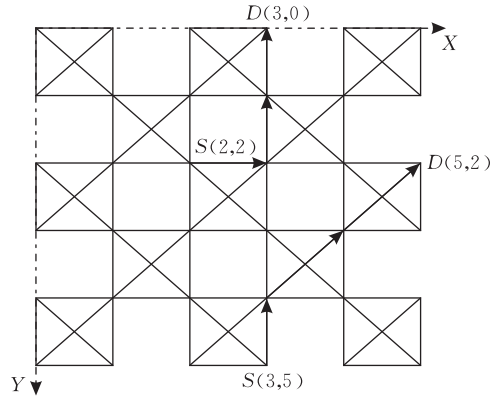
图 4 MIN 函数的伪代码

```
DR (cx,cy,dx,cy){
  if (cx==dx && cy==dy) HOME; //判断是否已到达目的
  else if (abs(cx-dx)+abs(cy-dy)==1) 走 X 或 Y 方向;
  //若与目标节点坐标距离为 1,直接走 X 方向或者 Y 方向
  else if (dx==0 && dy==0){ //若目标节点是 4 个顶点之一
    if (cx==1 && cy==1) 走 XY-方向;
    //若与目标节点在 XY 方向上距离为 1,走 XY 方向
    else MIN (cx,cy,1,1); //否则调用 MIN 函数
  }else if (dx==0 && dy==mesh_width-1){
    if (cx==1 && cy==mesh_width-2) 走 YX-方向;
    //若与目标节点在 XY 方向上距离为 1,走 YX 方向
    else MIN (cx,cy, 1, mesh_width-2);
                  //否则调用 MIN 函数,保证目标节点不在外围
  }else if (dx==mesh_width-1 && dy==0){
    if (cx==mesh_width-2 && cy==1) 走 YX+方向;
      else MIN(cx,cy,mesh_width-2,1);
  }else if (dx==mesh_width-1 && dy==mesh_width-1){
    if (cx==mesh_width-2 && cy==mesh_width-2)
      走 XY+方向;
    else MIN (cx,cy,mesh_width-2,mesh_width-2);
  }else if (dx==0){
    if (dy%2==1 && cx-dx==1 && dy-cy==1)
      走 YX-方向;
    else if (dy%2==0 && cx-dx==1 && cy-dy==1)
          走 XY+方向;
    else MIN (cx,cy,1,dy);
  }else if (dx==mesh_width-1){
    if (dy%2==1 && dx-cx==1 && dy-cy==1)
      走 XY+方向;
    else if (dy%2==0 && dx-cx==1 && cy-dy==1)
          走 YX+方向;
    else MIN (cx,cy,mesh_width-2,dy);
  }else if (dy==0){
    if (dx%2==0 && cx-dx==1 && cy-dy==1)
      走 XY-方向;
    else if (dx%2==1 && dx-cx==1 && cy-dy==1)
          走 YX+方向;
    else MIN(cx,cy,dx,1);
  }else if (dy==mesh_width-1){
    if (dx%2==0 && cx-dx==1 && dy-cy==1)
      走 YX-方向;
    else if (dx%2==1 && dx-cx==1 && dy-cy==1)
          走 XY+方向;
    MIN(cx,cy,dx,mesh_width-2);
  }else MIN(cx,cy,dx,dy);
}
```

图 5 DR 函数的伪代码



(a) 2层Rgrid的路由步骤



(b) 3层Rgrid的路由步骤

图 6

5 实验结果分析

5.1 模拟环境

使用龙芯用户级模拟器作为本文的模拟工具[14]. 该模拟器采用 CMP 结构,访存结构为分布式共享存储,各处理器核的二级 Cache 分别映射到不同的访存地址空间段,增大二级 Cache 空间,同时也带来更多核间通信开销. 该模拟器的特点还包括: L1 cache 和 L2 Cache 具有包含关系;使用基于目录的 Cache 一致性协议,目录包含在 L2 Cache 中;支持路由器使用多个虚通道;使用 Round Robin 的仲裁策略;使用基于 Credit 的流控策略,可选择使用虫孔或存储转发流控机制.

5.2 Benchmark

常用的计算机系统测试程序有 MiBench、Mediabench、DhryStone 等. 其中 MiBench 和 Mediabench 对应多媒体应用,DhryStone 是一种综合测试程序集,没有特别针对什么应用,比较均衡.

本文使用 Splash2 并行程序集作为测试程序,是 Stanford 大学开发的用于数值计算的基准测试程序,包含 8 个完整的应用程序和 4 个计算核心程序,都是科学与工程计算和计算机图形学方面的并行程序,主要用于评价 SMP、CC-NUMA、DSM 等共享存储类体系结构的计算机系统的性能. Splash2 的程序名分别为 Barnes、Cholesky、FFT、FMM、

LU、Ocean、Radiosity、Radix、Raytrace、Volrend、Water-Nsquared、Water-Spatial 等.

这些程序主要解决下列问题. Barnes：解决天体物理中的 n-body 问题；Fmm：模拟二维空间中物体运动；Water：水分子模拟；Cholesky：把一个稀疏矩阵分解成一个下三角矩阵与它的转置矩阵的乘积；Radix 是一个迭代开 r 次方的数学计算程序；Fft：计算迭代开 r 次方根的倒数；Lu：把一个密集矩阵分解成一个下三角矩阵和一个上三角矩阵的乘积.

文献[15]中给出了 Splash2 各程序通信量与处理器个数的关系：Radix 的通信量与处理器个数基本无关，只跟输入数据集的大小有关；Barnes、Fmm、Lu、Ocean、Water-sp 的核间通信量跟处理器数目的开平方成正比；Water-nsq 的通信量跟处理器数目成正比；Radiosity、Raytrace、Volrend、Cholesky 的通信量较难预测，跟数据的具体分布有关.

5.3　性能分析

由于 Splash2 中的个别程序在多核模拟器上的运行时间较长，这里选择 Splash2 中的 Ocean、Ocean_nc、Fft、Fmm、Lu、Lu_nc、Radix、Water_ns 等运行时间较短的程序进行分析，比较 Mesh 结构和 Rgrid 的性能优劣，其中 Mesh 结构使用 Dor 算法进行路由，Rgrid 使用上面提供的 DR 路由算法，Lu 与 Lu_nc 的区别，以及 Ocean 与 Ocean_nc 的区别主要在于输入数据的分配方式不同.

设处理器数为 16，Mesh 结构选用 DOR 算法进行路由，Rgrid 结构使用 DR 算法路由，在模拟器上运行 Splash2 中的 Ocean、Ocean_nc、Fft、Fmm、Radix、Lu、Lu_nc、Water_ns 等几个程序. 表 1 和图 7、图 8 是运行结果在两种拓扑结构中关于通信数量、平均延时以及 IPC 三方面的比较结果.

表 1　Mesh 和 Rgrid 结构上关于包总数的比较

	Mesh_packets	Rgrid_packets
Ocean	2097262	2100313
Ocean_nc	1726553	1731647
Fft	20345	20324
Fmm	37132	37374
Radix	1493483	1542272
Lu	7173096	7327666
Lu_nc	1493065	653339
Water_ns	2182218	2178038

每个程序运行过程中发送的数据包总量并非确定不变，等待应答的请求方如果在规定的时间内没有收到应答，请求方会重新发包，因此不同的运行环境对应的信息量是不等的，延时较小会相应减少重复发包的次数，从而减少数据包的总数，如表 1 中 Lu_nc 程序，在 Rgrid 结构中的数据包总数比 Mesh 结构的包总数少将近 50%. 各程序的通信数据包总

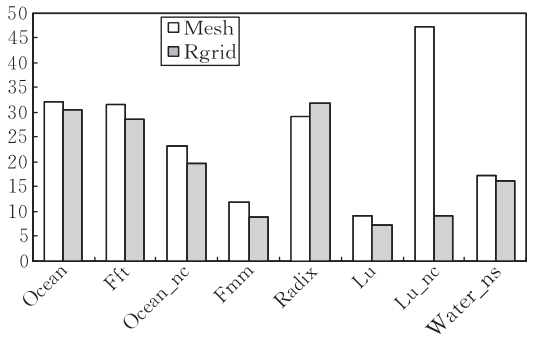


图 7　Mesh 与 Rgrid 结构关于平均延时的比较

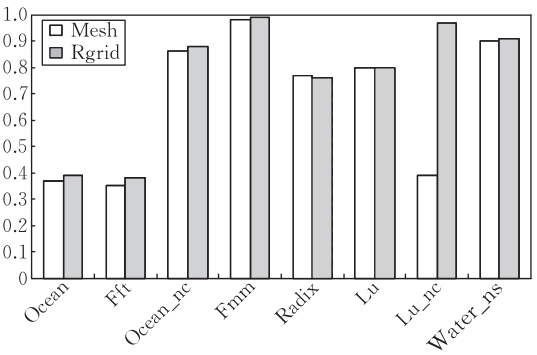


图 8　Mesh 与 Rgrid 结构关于 IPC 的比较

数与发送的密集程度不一致，延时的减小未必与 IPC 的增大成正比，但 IPC 会随着延时的减小而增大，或随着延时的增大而减小.

Radix 程序在 Mesh 结构中性能好些，原因在于 Rgrid 中有 4 对结点间的距离大于 Mesh 中的距离，如图 1 所示，这 4 对节点分别是：(0,1)和(0,2)，(3,1)和(3,2)，(1,0)和(2,0)，(1,3)和(2,3)，这 4 对节点距离较远的劣势中和了 Rgrid 总体节点间距离偏小的优势.

Lu_nc 程序的性能提高最明显，Rgrid 结构上所得 IPC 是在 Mesh 结构所得 IPC 的 2.48 倍，由此可见片间通信对于 Lu_nc 程序的性能影响很大. 究其原因，应该是 Mesh 结构中 Lu_nc 程序的某些数据包一直没有到达目标节点，造成处理器长时间等待响应，从而导致处理器利用率降低，这样的连锁反应导致它原来的 IPC 非常低，提高通信效率能够大幅度减小 Lu_nc 程序的执行时间，从而增大 IPC.

将 Rgrid 结构与 Mesh 结构得到的模拟结果作比较：Ocean 的 IPC 增加了 6%，平均延时降低了约 5.2%；Ocean_nc 的 IPC 增加了 8%，平均延时降低了 9.5%；Fft 的 IPC 增加了 2.7%，平均延时降低了约 15%；Fmm 的 IPC 增加了约 0.5，平均延时降低了 25%；Radix 的 IPC 减小了约 1.2%，平均延时提高了约 9%；Lu 的 IPC 增加了约 0.6%，平均延时降低了 20%；Lu_nc 的 IPC 增加了约 148%，平均延

时降低了 81%.

上述各个程序中平均延时的降低跟 IPC 的增加比例并不一致,原因在于各程序的通信负载总量和密集程度并不相同,因此通信延时的降低对总体性能的影响也不一致,但总的来说,Rgrid 结构所得 IPC 比 Mesh 结构提高了 0.6%～148%,Rgrid 所得平均延时比 Mesh 结构降低了约 5%～81%.

6 总 结

Rgrid 结构具有扩展性好、带宽高、节点间距小以及硬件实现难度低等优点.从模拟结果可以看出,对于 Splash2 中的大多数程序,Rgrid 结构性能比 Mesh 结构的性能优越,两种结构相比较,使用 Rgrid 结构时程序 IPC 最多可以提高 148%,平均延时可以降低 81%.目前在 C 语言实现的用户级多核处理器模拟器上进行实验分析,接下来计划详细分析这种结构的面积、功耗等其它影响具体实现的相关因素,最终将 Rgrid 结构应用到片上网络系统中.

参 考 文 献

[1] Sgroi M, Sheets M, Mihal A. Addressing the system-on-a-chip interconnect woes through communication-based design//Proceedings of the 38th Design Automation Conference. Las Vegas, NV, USA, 2001: 667-672

[2] Dally W J, Towles B. Route packets, not wires: On-chip interconnection networks//Proceedings of the Design Automation Conference (DAC). Las Vegas, NV, USA, 2001: 683-689

[3] Pande P P, Grecu C, Ivanov A, Saleh R. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. IEEE Transactions on Computers, 2005, 54(8): 1025-1040

[4] Farahabady M H, Sarbazi-Azad H. The WK-recursive pyramid: An efficient network topology//Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05). Las Vegas, NV, USA, 2005: 6

[5] Pavlidis Vasilis F, Friedman Eby G. 3-D topologies for networks-on-chip//Proceedings of the IEEE International SOC Conference. Austin, Texas, USA, 2006: 1081-1090

[6] Bartic T A, Mignolet J-Y, Nollet V, Marescaux T, Verkest D, Vernalde S, Lauwereins R. Topology adaptive network-on-chip design and implementation. IEE Proceedings of Computers and Digital Techniques, 2005, 152(4): 467-472

[7] Guerrier P, Greiner A. A generic architecture for on-chip packet-switched interconnections//Proceedings of the Design and Test In Europe (DATE). Paris, France, 2000: 250-256

[8] Kumar S et al. A network on chip architecture and design methodology//Proceedings of the International Symposium VLSI (ISVLSI). Pittsburgh, USA, 2002: 117-124

[9] Karim F et al. An interconnect architecture for networking systems on chips. IEEE Micro, 2002, 22(5): 36-45

[10] Pande P P, Grecu C, Ivanov A, Saleh R. Design of a switch for network on chip applications//Proceedings of the International Symposium on Circuits and Systems(ISCAS). Bangkok, Thailand, 2003, 5: 217-220

[11] Seo Daeho, Ali Akif, Lim Won-Taek, Rafique Nauman, Thottethodi Mithuna. Near-optimal worst-case throughput routing for two-dimensional mesh networks//Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA'05). Madison, Wisconsin, USA, 2005: 432-443

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

[13] Mukherjee S S, Bannon P, Lang S, Spink A, Webb D. The Alpha 21364 network architecture. IEEE Micro, 2002, 22(1): 26-35

[14] Huang Kun, Ma Ke, Zeng Hong-Bo, Zhang Ge, Zhang Long-Bing. A use-level simulator for tiled chip multiprocessor. Journal of Software, 2008, 19(4): 1069-1080(in Chinese)
(黄琨, 马可, 曾洪博, 张戈, 章隆兵, 一种分片式 CMP 的用户级模拟器, 软件学报, 2008, 19(4): 1069-1080)

[15] Woo S C et al. The SPLASH-2 programs: Characterization and methodological consideration//Proceedings of the 22nd Annual International Symposium on Computer Architecture. Santa Margherita Ligure, Italy, 1995: 24-36

ZHU Xiao-Jing, born in 1982, Ph. D., assistant researcher. Her research interests focus on computer architecture.

Background

The author now works in Institute of Computing Technology, taking part in the design of on-chip interconnection of Loongson No. 3. On-chip interconnection design includes topology design and router architecture design. The aim of this work is gaining a high performance and easy to realize on chip interconnection architecture. The topology choice affects the performance of on chip system seriously. So the author did many performance analyses, and designed many kinds of new topologies. The Rgrid topology is one of our topology design experiments.