用于自适应路由片上网络的缓冲分配算法

李晓辉[1,2]，曹 阳[1]，王力纬[1]，陈 晨[1]

(1. 武汉大学电子信息学院 武汉 430079; 2. 中国科学院高有物理研究所 北京 石景山区 100049)

【摘要】针对片上网络缓冲资源紧张的问题，提出了一种缓冲分配算法。在有限的资源下，该算法能够根据每个路由器输入通道上负载的情况来自动分配缓冲资源，从而获得最大的网络性能。在该算法中，提出了适用于自适应路由算法下的路由器性能分析模型，利用该模型可以快速定位系统中的性能瓶颈。仿真实验的结果表明，使用本算法后的NoC能比均匀分配策略下的NoC获得更小的数据包平均传输时延，同时，该算法还能节省约33%的缓冲资源。

关 键 词 自适应路由算法; 分析模型; 缓冲分配; 片上网络

中图分类号 TP393 文献标识码 A doi:10.3969/j.issn.1001-0548.2010.06.026

Buffer Allocation Algorithm for Adaptively-Routed Network-on-Chip

LI Xiao-hui[1,2], CAO Yang[1], WANG Li-wei[1], and CHEN Chen[1]

(1. School of Electronic Information, Wuhan University Wuhan 430079;
2. Institute of High Energy Physics, Chinese Academy of Sciences Shijingshan Beijing 100049)

Abstract For the intension of buffering resources in network-on-chip (NoC), a buffer allocation algorithm is proposed. Given buffering space budget, our algorithm automatically allocates the resources on each input channel, in different routers across the chip, to match the traffic load, such that the overall performance is maximized. In the algorithm, a novel analytical model for adaptive routing is used to quickly detect potential performance bottlenecks in the system. Simulation results indicate that our algorithm can get lower average packet latency than uniform allocation strategy, and about 33% savings in buffering resources can be achieved.

Key words adaptive routing algorithm; analytical model; buffer allocation; network-on-chip

随着半导体工艺的迅猛发展，一个单一的芯片上可集成几十个，甚至数百个处理单元(processing element，PE)和存储单元(storage element，SE)，为了解决各个单元间复杂的互连通信问题，提出了片上网络(NoC)[1-3]的概念。

1 片上网络

典型的片上网络由资源节点、路由器和双向链路组成。每个资源节点通过本地网络接口与一个路由器相连接，使各种不同的接口协议在网络层可以被屏蔽。每个路由器与相邻的4个路由器通过双向链路相连，数据主要以包的形式在各个资源节点间传输。在虫孔交换(wormhole switch)机制中，每个数据包被分为固定长度的微片。数据包的传输路径受路由算法的控制。

片上网络由于实现环境所限而面临严格的资源成本约束，为了降低实现代价，要求NoC的面积必须尽可能地小。在NoC路由器中，每个输入方向上的缓冲大小直接影响网络的性能。文献[4]的研究表明，路由器绝大部分面积都被输入缓冲所使用。因此，为了减小NoC面积而又不影响网络的性能，每个路由器输入缓冲的大小都必须仔细计算。

2 缓冲分配算法

2.1 问题描述

片上网络缓冲分配问题的实质就是在给定缓冲资源总量的前提下，如何在各个路由器输入通道间分配有限的资源，从而获得最大的网络性能。网络性能主要通过数据包平均传输时延L来衡量[4]，用数学方法可描述为：

已知缓冲资源的总量为B；数据包注入率为λ_g；

收稿日期：2009－04－13； 修回日期：2009－10－25
基金项目：国家863计划项目(2002AA1Z149)
作者简介：李晓辉(1982－)，男，博士生，主要从事NoC设计方法学方面的研究.

系统参数，如数据包头微片处理时间为H；数据包微片个数为M等，求每个输入方向上的缓冲大小$l_{x,y,\mathrm{dir}}$，使得数据包平均传输时延L最小，即：

$$\min(L)\ \text{s.t.}\ \sum_{\forall x}\sum_{\forall y}\sum_{\forall \mathrm{dir}} l_{x,y,\mathrm{dir}} \leqslant B$$

2.2 相关研究

传统的缓冲分配主要采用均匀分配策略，即为每个路由器分配相同大小的输入缓冲，但由于片上网络中业务流量分布不均匀，该方法无法获得最大的网络性能。更优的方法是在负载较重的输入通道上分配更多的缓冲资源，而负载较轻的通道上分配较少的资源。为了实现该分配方法，需要对NoC的性能进行评估，找出那些负载重的输入通道。文献[5]对2维网格(2D Mesh)结构进行了改进，提出了一种新的拓扑结构，在特定业务流量下取得了较好的网络性能。文献[6]提出了一种路由器分析模型，该模型适用于不同大小的输入缓冲和消息长度，但只对均匀分配方式有效。文献[7]提出了根据网络中的业务流量动态分配缓冲资源的方法，该方法需要对路由器的结构进行修改，增加了硬件开销。

文献[4]利用排队论建立了基于存储转发(store-and-forward)机制和虚跨步切换(virtual cut-through)机制的分析模型，并提出了一种缓冲分配算法。文献[8]对此做出改进，建立了用于虫孔交换机制下的分析模型。

文献[4]和文献[8]都仅考虑了确定性路由算法，然而在实际的应用中，自适应路由算法，特别是部分自适应路由算法，在片上网络中使用更为广泛[9]，因此本文将主要讨论如何在自适应路由算法下建立分析模型，并实现缓冲分配。

2.3 分析模型

由于不同拓扑结构下自适应路由算法各有区别，为使研究不失一般性，本文主要考虑2维网格结构[1-3]下的部分自适应路由算法(如DyAD routing、Turn-model based routing和Odd-even routing等)，这些算法实现简单，而且不需要使用虚通道。在此基础上，建立路由器分析模型，通过该模型，能够快速定位系统中的性能瓶颈。本文中，性能瓶颈被定义为所有输入通道中最可能出现满(full)的位置，即某一输入通道处缓冲为满的概率最大。

本文提出的分析模型基于如下假设[4, 6, 8, 11]：

(1) 网络中各节点发送数据包的过程相互独立，并且该过程满足泊松分布，平均注入率为每周期λ_g个数据包。此外，数据包的目的地址均匀地分布在网络中。

(2) 数据包的长度固定为M个微片，在没有阻塞的情况下，每个微片从一个路由器发送到相邻路由器所需要的时间为一个时钟周期。

(3) 路由器输入缓冲的宽度等于一个微片的位宽，因此输入缓冲容量可用微片个数来表示。

(4) 路由器在本地输入方向上的缓冲为无限大。此外，任何数据包到达目的节点后都被立即发送到处理单元。

根据排队论的相关知识[10]，可以把路由器东、西、南、北4个方向上的输入缓冲分别视为一个M/G/1/K的有限长队列，对于节点(x,y)，其dir方向上缓冲为满的概率为：

$$b_{x,y,\mathrm{dir}} = \frac{1-\rho_{x,y,\mathrm{dir}}}{1-\rho_{x,y,\mathrm{dir}}^{l_{x,y,\mathrm{dir}}+1}}\rho_{x,y,\mathrm{dir}}^{l_{x,y,\mathrm{dir}}} \tag{1}$$

其中：

$$\rho_{x,y,\mathrm{dir}} = \frac{\lambda_{x,y,\mathrm{dir}}}{\mu_{x,y,\mathrm{dir}}} = \lambda_{x,y,\mathrm{dir}}T_{x,y,\mathrm{dir}} \tag{2}$$

因此，只要求得路由器每个输入通道的数据包到达率$\lambda_{x,y,\mathrm{dir}}$和平均服务时间$T_{x,y,\mathrm{dir}}$，就能够计算出对应的概率值，并由此找到系统的性能瓶颈对其进行优化。

假设a和b是网络中两个相邻的节点，而s和d分别为源节点和目的节点，由文献[6]可知，对于一对特定的源、目的节点(s,d)，其数据包传输路径经过通道$\langle a,b\rangle$的概率可以表示为：

$$P_{(s,d),\langle a,b\rangle} = \frac{\dfrac{(|x_s-x_a|+|y_s-y_a|)!}{|x_s-x_a|!\times|y_s-y_a|!}\ \dfrac{(|x_d-x_b|+|y_d-y_b|)!}{|x_d-x_b|!\times|y_d-y_b|!}}{\dfrac{(|x_s-x_d|+|y_s-y_d|)!}{|x_s-x_d|!\ |y_s-y_d|!}} \tag{3}$$

式中 x_s、y_s分别为各节点在网络中的坐标值。

由假设(1)可知，网络中的流量为均匀随机流量，则通道$\langle a,b\rangle$上的数据包到达率为：

$$\lambda_{(s,d),\langle a,b\rangle} = \frac{\lambda_g}{N-1}P_{(s,d),\langle a,b\rangle} \tag{4}$$

式中 N表示网络中的节点数，对于$K\times K$的网络，$N=K^2$；λ_g为每个节点的包注入率。

当网络中有多对源、目的节点时，该通道上总的数据包到达率为：

$$\lambda_{\langle a,b\rangle}=\frac{\lambda_g}{N-1}\sum_{(s,d)\in G_{\langle a,b\rangle}}P_{(s,d),\langle a,b\rangle} \tag{5}$$

式中　$G_{\langle a,b\rangle}$是由多对源、目的节点组成的集合，在该集合中，每对节点至少有一条最短路径经过通道$\langle a,b\rangle$。

在单个节点中，可以根据式(5)求出数据包从某一输入端口传输到另一输出端口时的速率。对于一对源、目的节点(s,d)，假设$p_{(s,d),\langle a,b\rangle\to\langle b,c\rangle}$是数据包从节点$b$中与通道$\langle a,b\rangle$相连的输入端口传输到与通道$\langle b,c\rangle$相连的输出端口的概率，则只需要将式(3)中$b$的坐标$(x_b,y_b)$替换为$c$的坐标$(x_c,y_c)$即可求出其值。若将与$\langle a,b\rangle$相连的那个输入端口设为$i$，而与$\langle b,c\rangle$相连的输出端口设为$j$，则数据包传输路径经过通道$\langle b,c\rangle$的概率可表示为$p_{(s,d),(x_b,y_b),(i\to j)}$。设数据包从$i$发送到$j$的速率为$\lambda_{x_b,y_b,i\to j}$，由式(5)有：

$$\lambda_{x_b,y_b,i\to j}=\frac{\lambda_g}{N-1}\sum_{(s,d)\in G(i\to j)}p_{(s,d),(x_b,y_b,(i\to j)} \tag{6}$$

式中　$G_{(i\to j)}$是多对源、目的节点组成的集合，在该集合中，每对节点至少有一条最短路径经过节点b，且由i输入，从j输出。

在节点(x,y)中，若用dir和dir′分别代表输入和输出端口所对应的方向，则$\lambda_{x,y,i\to j}$可表示为$\lambda_{x,y,\mathrm{dir}\to\mathrm{dir}'}$，dir方向上数据包到达率为：

$$\lambda_{x,y,\mathrm{dir}}=\sum_{\mathrm{dir}'\in G_{(\mathrm{out})}}\lambda_{x,y,\mathrm{dir}\to\mathrm{dir}'} \tag{7}$$

式中　$G_{(\mathrm{out})}$是所有输出方向的集合。对于网络中的边界节点，由于在某些方向上未与任何节点相连，因此不会有数据包到达，$\lambda_{x,y,\mathrm{dir}}=0$。

节点(x,y)在dir方向上数据包的平均服务时间可以表示为：

$$T_{x,y,\mathrm{dir}}=H+M+B_{x,y,\mathrm{dir}} \tag{8}$$

式中　H为数据包头微片处理时间；M为数据包的微片个数；$H+M$为无阻塞情况下数据包的服务时间；$B_{x,y,\mathrm{dir}}$为数据包在该输入方向上的阻塞时间，可由数据包在该方向上的阻塞概率$Pb_{x,y,\mathrm{dir}}$与平均等待时间$W_{x,y,\mathrm{dir}}$求得：

$$B_{x,y,\mathrm{dir}}=Pb_{x,y,\mathrm{dir}}W_{x,y,\mathrm{dir}} \tag{9}$$

平均等待时间$W_{x,y,\mathrm{dir}}$是数据包因阻塞而等待的时间，由于路由器的每个输入缓冲都被看作是一个M/G/1/K的有限长队列，根据排队论[10]知：

$$W_{x,y,\mathrm{dir}}=\frac{\rho S(1+C_s^2)}{2(1-\rho)} \tag{10}$$

式中　$\rho=\lambda S$；$C_S^2=\dfrac{\sigma^2}{S^2}$；$\lambda$为该队列的包到达率，在本文中即为$\lambda_{x,y,\mathrm{dir}}$；$S$为数据包平均服务时间，即$T_{x,y,\mathrm{dir}}$；$\sigma^2$为$S$的概率方差，由文献[11]知，$\sigma^2$可近似为$(T_{x,y,\mathrm{dir}}-M)^2$。将上述参数代入式(10)有：

$$W_{x,y,\mathrm{dir}}=\frac{\lambda_{x,y,\mathrm{dir}}T_{x,y,\mathrm{dir}}^2\left[1+\dfrac{(T_{x,y,\mathrm{dir}}-M)^2}{T_{x,y,\mathrm{dir}}^2}\right]}{2(1-\lambda_{x,y,\mathrm{dir}}T_{x,y,\mathrm{dir}})} \tag{11}$$

在虫孔交换机制的NoC中，数据包的阻塞可分为两类：一类是因多个数据包对同一输出通道竞争而引起的阻塞，设其概率为$\theta_{x,y,\mathrm{dir}}$；另一类是因路由器的输入通道缓冲资源不足而导致的阻塞，设其概率为$\gamma_{x,y,\mathrm{dir}}$。综合上述两种阻塞类型，数据包总的阻塞概率可表示为：

$$Pb_{x,y,\mathrm{dir}}=\theta_{x,y,\mathrm{dir}}\times\gamma_{x,y,\mathrm{dir}} \tag{12}$$

为了求得$\theta_{x,y,\mathrm{dir}}$和$\gamma_{x,y,\mathrm{dir}}$，需要计算出节点$(x,y)$的一个5×5的转发概率矩阵$\boldsymbol{F}$，其每个元素为：

$$f_{\mathrm{dir}\to\mathrm{dir}'}=\frac{\lambda_{x,y,\mathrm{dir}\to\mathrm{dir}'}}{\lambda_{x,y,\mathrm{dir}}} \tag{13}$$

式中　$\lambda_{x,y,\mathrm{dir}\to\mathrm{dir}'}$和$\lambda_{x,y,\mathrm{dir}}$可分别由式(6)和式(7)计算得到。转发概率$f_{\mathrm{dir}\to\mathrm{dir}'}$表示数据包从dir方向转发到dir′方向的概率，当dir与dir′相同时，其值为0。

根据转发概率矩阵$\boldsymbol{F}$可求得数据包从dir方向转发到dir′方向时被阻塞的概率$C_{\mathrm{dir}\to\mathrm{dir}'}$：

$$C_{\mathrm{dir}\to\mathrm{dir}'}=f_{\mathrm{dir}\to\mathrm{dir}'}\sum_{\forall k}f_{k\to\mathrm{dir}'} \tag{14}$$

并有：

$$\theta_{x,y,\mathrm{dir}}=\sum_{\forall\mathrm{dir}'}f_{\mathrm{dir}\to\mathrm{dir}'}C_{\mathrm{dir}\to\mathrm{dir}'} \tag{15}$$

由假设(4)可知，数据包到达目的节点后能立即被处理单元所接收，因此在该情况下不会发生数据包阻塞，$\gamma_{x,y,\mathrm{dir}}$就仅与相邻路由器输入缓冲的状态有关。为便于分析，假设此时dir为东向(dir为E)，而其他输入方向可采用相同的方式求得，则有：

$$\gamma_{x,y,\mathrm{E}}=f_{\mathrm{E}\to\mathrm{W}}b_{x-1,y,\mathrm{E}}+f_{\mathrm{E}\to\mathrm{N}}b_{x,y+1,\mathrm{S}}+f_{\mathrm{E}\to\mathrm{S}}b_{x,y-1,\mathrm{N}} \tag{16}$$

式中　$b_{x,y,\mathrm{dir}}$为输入缓冲为满的概率，同式(1)。

将式(15)和式(16)代入式(12)中求得阻塞概率$Pb_{x,y,\mathrm{dir}}$的表达式，并结合式(8)～式(11)可得到关于$T_{x,y,\mathrm{dir}}$的非线性方程，联立NoC中每个输入通道上的非线性方程组成方程组，求解该方程组就能得到数据包的平均服务时间$T_{x,y,\mathrm{dir}}$。将计算出的$T_{x,y,\mathrm{dir}}$和$\lambda_{x,y,\mathrm{dir}}$代入式(1)可求得不同输入方向上缓冲为满的概率$b_{x,y,\mathrm{dir}}$，通过比较，即可确定系统的性能瓶颈所在。

2.4 算法步骤

基于上述分析模型，本文采用了一种贪婪的缓冲分配算法[8]，它能够向负载最重的输入通道自动分配更多的缓冲资源，具体步骤如下：

(1) 初始分配。假设所有有效输入通道($\lambda_{x,y,\mathrm{dir}}\neq 0$)都被分配了一个微片大小的缓冲单元，即$l_{x,y,\mathrm{dir}}=1$。同时，向分配算法的程序(在Matlab中实现)中输入相关的参数，包括系统参数(如H、M等)和业务参数(如λ_g等)，根据式(7)和式(13)求得$\lambda_{x,y,\mathrm{dir}}$及$\boldsymbol{F}$，结合式(8)～式(16)，可得到关于$T_{x,y,\mathrm{dir}}$的非线性方程组。

(2) 求解方程组。调用Matlab的fsolve函数求解关于$T_{x,y,\mathrm{dir}}$的非线性方程组，将所得结果代入式(1)求出每个输入方向上缓冲为满的概率。

(3) 结果比较。比较步骤(2)中求得的各概率值，并判断出最大值所对应的输入通道。

(4) 重新分配。根据步骤(3)得到的结果，为对应方向的输入通道增加一个微片大小的缓冲单元，缓冲资源总量B相应地减少。

(5) 重复步骤(2)～步骤(4)，直至总的缓冲资源都被分配完(即B=0)。

(6) 输出最终分配结果。

该分配方式虽然比较简单，但从下面的实验结果看，能够取得比较好的分配效果。

3 仿真结果与分析

为验证本文所提出的算法，利用System C[12]构建一个NoC性能仿真平台，其具有高时钟模拟精度以及良好的可扩展性，可以对网络大小、拓扑结构、路由算法等灵活配置。仿真采用数据包微片个数M=16，数据包头微片处理时间为两个时钟周期(H=2)，网络结构为4×4 Mesh。基于转弯模型的北向最后(north-last，NL)路由算法和DyAD路由算法[13-14]，在一种典型的业务流量，即均匀随机流量(uniform)情况下进行。在该流量下，每个节点都以相同的概率向任意节点发送数据包。每次仿真过程都持续5×10^5个时钟周期，为避免采集到网络不稳定时的数据，前1×10^5个周期属于预热阶段，数据包信息不予采用。在仿真结果中，用Uniform N代表为每个输入通道分配N个缓冲单元的均匀分配算法，而用Customized N代表相同缓冲资源下本文所提出的分配算法。

图1表示了NL路由算法下不同分配算法的仿真结果。从图中可以看出，在较低的注入率下(<0.012 packet·cycle^{-1})，各分配算法的网络性能基本相同。随着注入率的增加，Uniform 4和Uniform 6先后因输入通道缓冲资源不足导致网络中发生数据包拥塞，整个网络性能迅速趋于饱和。而Customized 4预先通过分析模型估算出了系统的性能瓶颈并向其分配了较多的缓冲资源，有效地缓解了该处数据包的拥塞，最终获得了较小的数据包时延。由实验结果可知，Customized 4的性能要优于Uniform 6，说明通过本文的分配算法可为NoC节省约33%的缓冲资源达到比均匀分配算法更好的性能。

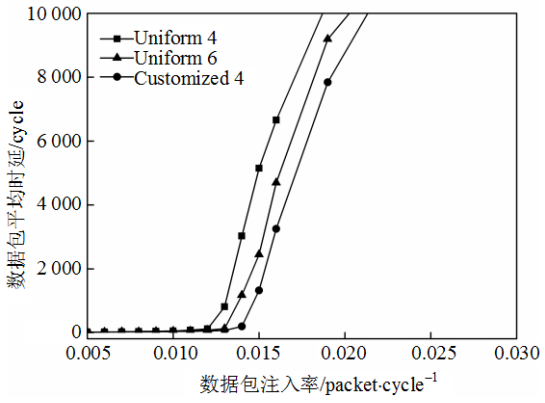

图 1 NL 路由算法下的性能曲线

图2为DyAD路由算法下的仿真结果。可以看到，此时Customized 4的性能依然优于其他算法，且其饱和吞吐率为0.013 packet·cycle^{-1}，比Uniform 6提高了约18.2%。而在图1中，Customized 4的饱和吞吐率仅比Uniform 6提高了$(0.014-0.013)/0.013\approx 7.7\%$。本文算法在DyAD路由算法下能获得更好的分配效果，这主要是因为，与DyAD路由算法相比，NL路由算法在一定程度上使网络中业务流量的分布更均匀，从而减少了系统中可能存在的性能瓶颈。在该情况下，均匀分配算法可以取得相对较好的分配效果，而本文算法对系统性能的优化作用则不明显。实际上，网络中业务流量分布越不均匀，系统中潜在的性能瓶颈就越多，本文算法对系统的优化程度就越高，反之就越低。

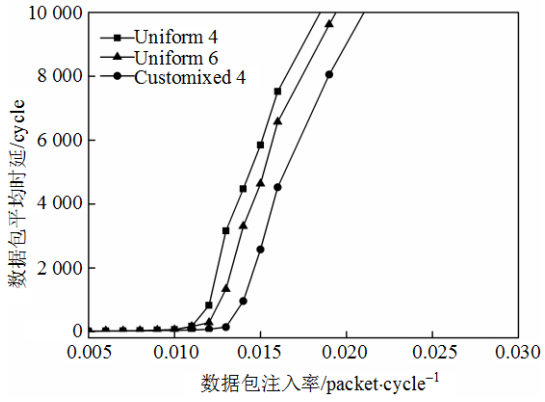

图 2 DyAD 路由算法下的性能曲线

为进一步验证本文算法的性能，将该算法与文献[4]中的线性启发式算法进行比较。启发式算法的基本思想是利用路由器输入通道上负载大小与缓冲资源间的线性比例关系进行分配。在图3中，线性启发式算法(linearly proportional，LP)用LP *N*表示，路由算法为DyAD路由算法。可以看到，当缓冲资源总量相同时，线性启发式算法具有比均匀分配算法更好的性能，但是与本文算法相比，两者数据包时延仍相差较大，即本文算法的分配效果要优于文献[4]中的线性启发式算法。

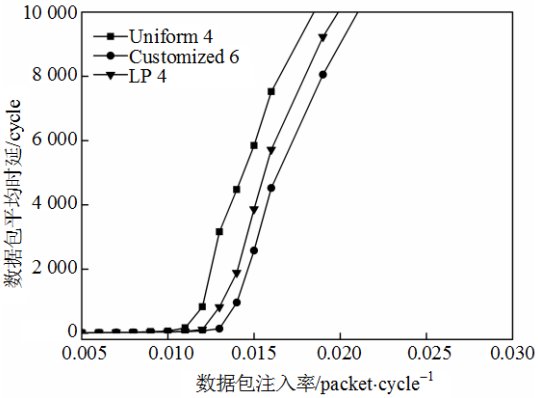

图 3　不同算法与线性启发式算法比较的性能曲线

4　总　结

本文针对网格型NoC中的部分自适应路由算法，建立了路由器性能分析模型，并以此为基础，提出了一种缓冲分配算法。该算法首先根据分析模型判断出系统中的性能瓶颈，随后通过向其分配较多的缓冲资源获得网络性能的提升。仿真结果表明，经过本文算法的分配，有效地利用了已有的缓冲资源，降低了数据包平均传输时延。未来的研究工作将主要集中在对现有模型的扩展和获得更精确的分配结果上。

参　考　文　献

[1] KUMAR S, JANTSCH A, SONININEN J P, et al. A network on chip architecture and design methodology[C]// Proceedings of the IEEE Computer Society Annual Symposium on VLSI. Pennsylvania: IEEE Press, 2002: 117-124.

[2] BENINI L, DEMICHELI G. Networks on chips: a new SoC paradigm[J]. Computer, 2002, 35(1): 70-78.

[3] HEMKEL J, WOLF W, CHAKRADHAR S. On-chip networks: A scalable, communication centric embedded system design paradigm[C]//Proceedings of the IEEE International Conference on VLSI Design. Mumbai: IEEE Press, 2004: 845-851.

[4] HU J, OGRAS U Y, MARCULESCU R. System-level buffer allocation for application specific networks -on-chip router design[J]. IEEE Transaction on Computer-aided Design of Integrated Circuits and Systems, 2006, 25(12): 2919-2933.

[5] ANJUM S, CHEN Jie, YUE Pei-pei, et al. Delay optimized architecture for on-chip communication[J]. Journal of Electronic Science and Technology of China, 2009, 7(2): 104-109.

[6] BAHN J H, BAGHERZADEH N. Design of simulation and analytical models for a 2d-meshed asymmetric adaptive router[J]. IET Computer Digital Technology, 2008, 2(1): 63-73.

[7] 赖明澈, 王志英, 郭建军, 等. 具有拥塞缓解策略的动态虚拟通道研究及其VLSI实现[J]. 计算机学报, 2008, 31(11): 2026-2037.
LAI Ming-che, WANG Zhi-ying, GUO Jian-jun, et al. Research and VLSI implementation of a dynamic virtual channel structure with congestion awareness scheme[J]. Chinese Journal of Computers, 2008, 31(11): 2026-2037.

[8] 王力纬, 曹　阳, 李晓辉, 等. 虫孔路由NoC的缓冲分配算法[J]. 北京邮电大学学报, 2008, 31(4): 29-32.
WANG Li-wei, CAO Yang, LI Xiao-hui, et al. A buffer allocation algorithm for wormhole routing networks-on-chip[J]. Journal of Beijing University of Posts and Telecommunications, 2008, 31(4): 29-32.

[9] BARATI H, MOVAGHAR A, BARATI A, et al. Routing algorithms study and comparing in interconnection networks[C]//Proceedings of 3rd International Conference on Information and Communication Technologies. Damascus: IEEE Press, 2008: 1-5.

[10] 盛友招. 排队论及其在现代通信中的应用[M]. 北京: 人民邮电出版社, 2007.
SHENG You-zhao. Queuing theory and its application in modern communications [M]. Beijing: Posts and Telecommunications Press, 2007.

[11] PATOOGHY A, SARBAZI-AZAD H. Analytical performance modeling of partially adaptive routing in wormhole hypercube[C]//Proceedings of 20th International Parallel and Distributed Processing Symposium. Rhodes Island: IEEE Press, 2006: 1-7.

[12] Open System C Initiative. System C version 2.0 user's guide[DB/OL]. [2002-02-08]. http:// www.systemc. org.

[13] DUATO J, YALAMANCHILI S, LIONEL M. Interconnection networks—an engineering approach[M]. San Francisco: Morgan Kaufmann Publisher, 2003.

[14] HU J, MARCULESCU R. DyAD-smart routing for networks-on-chip[C]//Proceedings of the 41st Design Automatic Conference. San Diego: ACM Press, 2004: 260-263.

编　辑　张　俊