文章编号: 1001- 2486(2009) 05- 0044- 06

一种面向应用的 NOC 缓冲区分配算法*

尹亚明, 陈书明, 孙书为, 王耀华
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 片上互连网络是片上通信问题的有效解决方案, 但其存在严重的资源限制。输入缓冲区占据片上网络总面积的显著部分, 同时其容量大小对不同应用映射后获得的性能有重要影响。给出一种面向应用数据负载的 NOC 缓冲区分配算法, 针对不同的应用映射, 该算法可以根据数据流量分布特征实现各个路由器输入通道上缓冲区资源的定制分配。实验结果表明, 使用该算法后, 系统缓冲区资源得到了更有效的利用。与均匀分配缓冲区的 NOC 系统相比, 采用该算法实现的缓冲区分配方案使系统在保持性能变化不大的情况下, 能够节省约 50% 的缓冲区总容量。

关键词: 片上网络; 缓冲区分配; 数据负载; 流量特征
中图分类号: TP302 文献标识码: A

An Application-specific Buffer Allocation Algorithm for Network-on-chip

YIN Ya-ming, CHEN Shu-ming, SUN Shu-wei, WANG Yao-hua
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The networks-on-chip(NOC) approach was proposed as a promising solution to on-chip communication problems, but it is by far more resource limited. The input buffers in a typical on-chip router take a significant portion of the silicon area of NOC, and the performance of a NOC is drastically affected by the amount of buffering resources. In this paper, an application-specific buffer planning algorithm that can be used to customize the router design in NOC is presented. More precisely, given the mapping of the target application and the traffic characteristics, the algorithm automatically assigns the buffer depth for each input channel, in different routers across the chip. The experimental results show that the system buffering resources can be utilized more effectively. In contrast with the uniform buffer allocation, about 50% saving in buffering resources can be achieved by automatic buffer allocation using our algorithm without any reduction in performance.

Key words: network-on-chip; buffer allocation algorithm; data payload; traffic characteristic

在半导体工艺技术不断推动下, 单颗芯片上能够集成数十个甚至更多带有大量嵌入式存储体的 IP 模块。丰富的片上计算资源(CPU 或 DSP, 视频处理器等) 对系统的片上通信资源提出了极大的需求。

片上互连网络(NOC) 成为片上通信问题的有效解决方案[1-4]。在一个 NOC 体系结构中, 芯片被分成一系列互连模块或节点, 每个节点可以是通用处理器、DSP 或存储子系统等。图 1 所示是采用二维 mesh 拓扑结构连接节点构成的一个典型 NOC 系统实例, 路由器嵌入每个节点负责实现与相邻节点的连接。用于 2-D mesh NOC 的路由器具有东、西、南、北四个端口和一个本地端口, 每个端口带有相应输入缓冲区, 本地端口与处理节点连接。

与标准的数据宏观网络相比, 片上互连网络存在更严重的资源限制。为达到系统实现代价的最优化, 片上互连网络实现应尽可能地降低其面积开销。在一个典型的片上路由器中, 输入缓冲区占据了片上互连网络总面积的显著部分[5]。在基于报文交换的网络中, 路由器的处理逻辑仅占整个路由器面积的 6.6%[2]。因此应在设计过程中精心考虑输入缓冲区容量的最小化问题。另一方面, 不同的应用程

* 收稿日期: 2009- 07- 03
基金项目: 国家 863 计划资助项目(2007AA01Z108); 教育部"高性能微处理器技术"创新团队资助项目(IRT0614); 国家自然科学基金资助项目(60676010)
作者简介: 尹亚明(1979—), 男, 博士生。

序之间的数据流量特征存在显著差异, 因此在系统设计之初要精心分配每个输入通道的缓冲区资源, 以达到与特定应用通信模式相匹配的目的。在现有 NOC 设计中被广泛使用的统一均匀分配缓冲区的方法不能达到上述目的, 从而会导致性能降低或增加额外面积开销。针对此类问题, 本文提出了一种面向应用数据负载的缓冲区分配算法, 实现不同路由器通道上缓冲区资源的定制分配, 以达到与目标应用通信流量特征相匹配的目的。

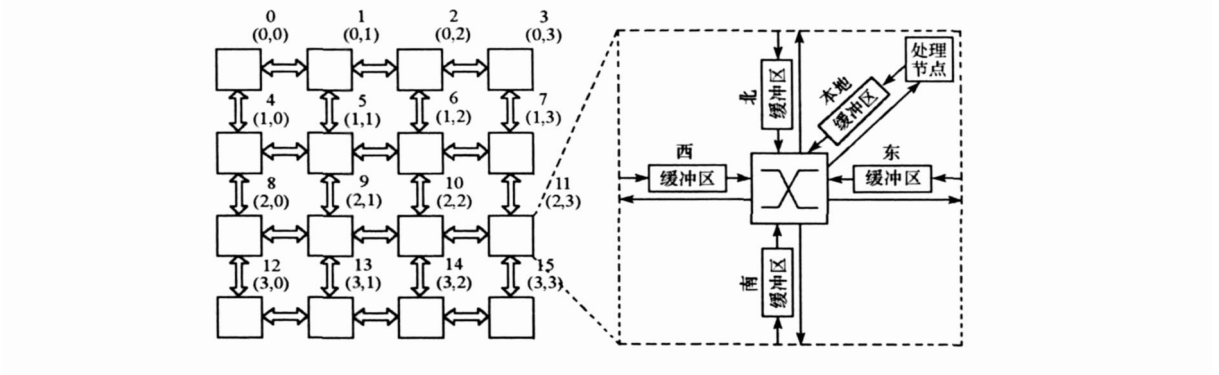

图 1 4× 4 mesh 结构 NOC 示例

Fig. 1 Block diagram of 4× 4 mesh-based NoC

1 相关研究

典型的 NOC 设计通常会面向某个或某一类特定的应用。因此, NOC 体系结构可针对目标应用进行定制化设计, 从而达到最佳的性能与成本折中[1,6]。文献[7– 8]指出针对系统面积、功耗和延迟实现的网络拓扑结构定制所获得的好处。文献[9– 10]研究了 IP 模块到 NOC 体系结构的拓扑映射以获得带宽和通信代价的节省。

本文工作面向特定应用实现每个路由器不同通道上的缓冲区资源定制分配。传统并行计算的性能评价使用耗时的仿真[11– 12]或给出用于受限流量特征下解析模型(典型的是均匀模型)。由于互连结构要能够支持更广泛的应用范围, 均匀流量特征的假设对于通用并行计算更有意义。然而, 这样的假设不再适用于 NOC 设计。典型 NOC 设计面向特定应用, 这样的应用将显现出特有的流量模型。

文献[5, 13]对高效的 NOC 缓冲区结构设计问题进行了研究。在文献[13]中, 以面积和性能为出发点, 采用硬件定制的 FIFO 结构。文献[5]给出 Proteo NOC 的缓冲区实现。但它们均未考虑如何定制缓冲区容量以匹配应用流量特征的问题。

文献[16]在不影响系统性能的情况下, 采用定制虚通道的方法节省了 40% 的缓冲区空间。但其针对应用流量特征提取部分工作并不充分。文献[15]给出一种拥塞感知的动态分配虚通道的结构, 实现虚通道在高低流量下的不同使用模式, 但其实现复杂, 并且要额外设计拥塞感知逻辑。

文献[16]首次系统地提出并解决面向应用的缓冲区分配问题, 文中基于排队论建立了适用于存储转发和虚跨步机制的路由器分析模型, 基于该模型提出一种缓冲区分配算法。文献[17]在文献[16]的基础上研究了虫孔路由 NOC 中的缓冲区分配问题。文献[16– 17]都是基于排队论构建路由器缓冲区排队模型, 通过解析与近似的方法获得非线性方程组以用来计算缓冲区满的概率, 从而作为分配缓冲区容量的依据。但这种方法存在理论近似性, 因为模型的精确度以及计算过程可能会对最终算法结果造成很大的影响。本文从实际应用流量特征模型出发, 分析获得数据流量负载特征分布情况, 提出一种面向应用数据负载的缓冲区分配算法。同时, 文献[16– 17]中的算法执行过程都是给定缓冲区总容量, 算法执行完毕获得面向应用的缓冲区分配方案。而本文中的算法根据设计约束给定的缓冲区容量最大最小值进行计算, 针对应用数据流量特征分布情况来调整这两个值, 可使得算法执行完毕获得的缓冲区分配方案能够更加有效地使用缓冲区空间。

2 缓冲区分配算法

在给定节点连接拓扑结构和应用映射条件下,算法可自动根据不同数据通路上的数据负载情况,对每个路由器输入端口的缓冲区容量进行分配,从而给出最终的缓冲区分配方案。

2.1 问题描述

文中使用的参数符号定义见表 1。依据给出参数,面向应用数据负载的缓冲区分配问题可描述为:

已知条件:

(1) 系统拓扑结构连通图 $G(S, R)$。其中 S 为系统节点标号集合,$s_i \in S$,s_i 为节点 i 标号。R 为数据通路集合,$r_{ij} \in R$ 为从节点 s_i 到节点 s_j 的有向连接,也可记为$(s_i, s_j) \in R$。

(2) 应用映射后通路上的数据流负载集合 D。集合 D 的元素与 R 的元素之间存在一一对应关系,即对于 $r_{ij} \in R$ 有 $d_{ij} \in D$ 与之对应,d_{ij} 为通路(s_i, s_j)上的数据负载。

(3) 定义:带数据负载的数据通路集合 E,对于 $e_{ij} \in E$,由三个元素构成(s_i, s_j, d_{ij}),s_i 为通路起点,s_j 为通路终点,d_{ij} 为应用映射后的通路数据负载。

(4) 根据数据流量特征与具体实现约束给出的缓冲区分配的上下限 L_{max} 和 L_{min}。通常 L_{min} 可设置为 1,但根据系统设计需求和资源情况可设置其他均值。L_{max} 则要根据设计目标和系统资源总量情况以及数据流量分布特征进行确定。

目标:

获得每个路由节点输入端口的缓冲区容量分配方案,针对带数据负载的数据通路集合 E,算法执行完毕生成对应带缓冲区容量的数据通路集合 E',对于 $e'_{ij} \in E'$,由三个元素构成(s_i, s_j, l_{ij}),s_i 为通路起点,s_j 为通路终点,l_{ij} 为算法执行后获得的缓冲区容量。该容量对应位置为节点 s_j 与通路(s_i, s_j)相连的输入端口。

表 1 参数定义

Tab. 1 Parameter notation

参数	描述
S	系统中节点集合
R	系统拓扑中数据通路集合
D	应用映射后,数据通路上数据负载集合
E	带数据负载的数据通路集合
E'	带缓冲区容量分配方案的数据通路集合
s_i	节点 i 的标号
r_{ij}	从节点 s_i 到节点 s_j 的有向连接通路
d_{ij}	数据通路 r_{ij} 上的数据负载
l_{ij}	节点 s_j 的路由器与节点 s_i 相连的输入端的缓冲区容量
e_{ij}	(s_i, s_j, d_{ij}) 带数据负载的数据通路元素
e'_{ij}	(s_i, s_j, l_{ij}) 带缓冲区容量分配方案的数据通路元素
L_{max}	缓冲区容量分配预定最大值
L_{min}	缓冲区容量分配预定最小值,默认为 1

2.2 算法步骤

如图 2 所示为本文给出的面向应用数据负载缓冲区分配算法的执行流程,采用 C++ 语言实现。算法执行过程描述如下:

步骤 1:输入系统参数(拓扑结构数据通路集合 R,应用映射后数据通路数据负载集合 D)和设计限定参数(缓冲区容量上下界 L_{max} 和 L_{min}),算法程序根据 R 和 D 的一一对应关系求得带流量分布的数据通路集合 E,同时给出依据缓冲区容量下界 L_{min} 获得的初始缓冲区分配方案(每个数据通路都按 L_{min} 进

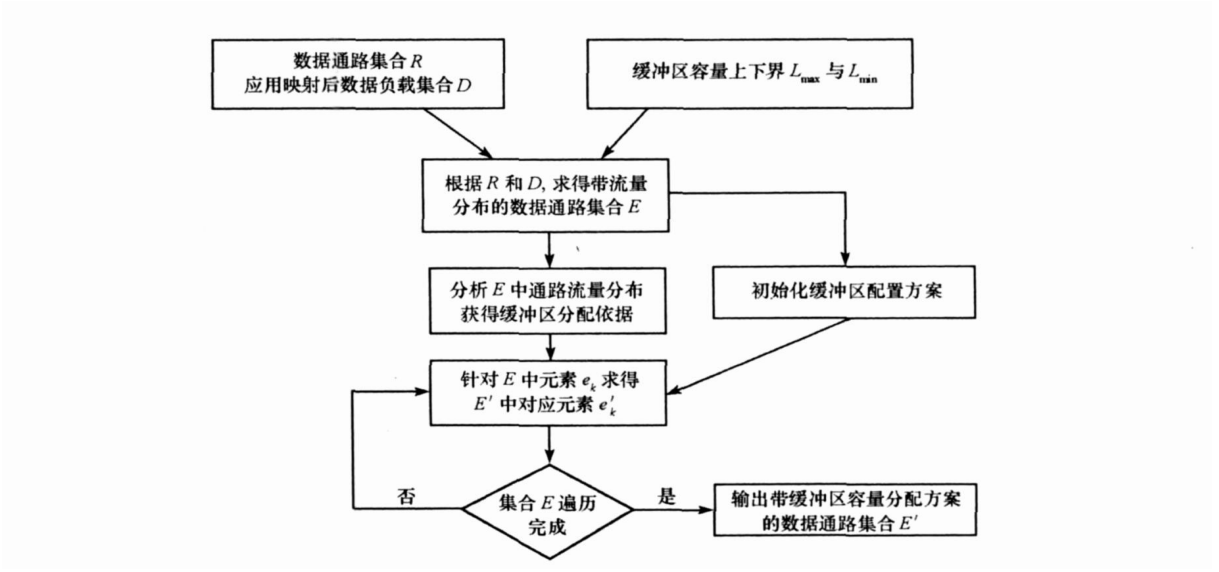

图 2 算法执行流程图

Fig. 2 Buffer allocation algorithm flow

行缓冲区分配) 。

步骤 2: 分析集合 E 中各通路的流量分布特征, 获得数据通路上数据负载的最大最小值, 并统计不同数据负载在所有数据通路中的分布情况和所占比例信息。根据系统设计限定参数 L_{max} 和 L_{min}, 分别求出不同的数据负载对应的缓冲区容量, 这也就是各输入端口缓冲区容量分配的依据。

步骤 3: 在已有初始化缓冲区分配方案基础上, 针对集合 E 中的每个元素, 根据步骤 2 中求得的缓冲区分配依据逐个计算集合 E' 中元素(s_i, s_j, l_{ij}) 的 l_{ij} 值。

步骤 4: 重复步骤 3, 直至系统中集合 E 遍历完成, 至此实现对系统中每个路由节点输入端口的缓冲区分配。

步骤 5: 输出集合 E' 作为缓冲区算法执行最终结果。集合 E' 中对应数据通路上的 l_{ij} 分量即为该通路输入端口的缓冲区分配方案。

在算法执行的 5 个步骤中, 步骤 2 为算法计算缓冲区分配方案的关键。其目标就是从已知应用映射后的流量特征分布和缓冲区上下界参数 L_{max} 和 L_{min} 获得不同流量特征的数据通路上对应的缓冲区容量。

具体说, 考虑 4×4 的 2-D mesh 网络, 若其映射后的数据流量分布范围为$\{0\sim 10\}$(其含义为单位时间传输的数据量, 此处仅作为举例说明), 根据系统设计目标和资源约束给出的缓冲区分配可取范围$\{1, 2, 4, 6, 8\}$(其含义为缓冲区数据单元个数) 。对于每条数据通路的输入端, 其缓冲区容量可在所给 5 种取值范围内进行分配。构造映射函数 $y=f(x)$, 其中 x 为数据流量值, y 则为该数据流量对应的缓冲区分配方案。在上述事例中, 可定义映射函数为$f(x)=\begin{cases}1, & 0<x\leqslant 2\\ \lceil \frac{x}{2}-1\rceil\times 2, & 2<x\leqslant 10\end{cases}$, 对于$\{0\sim 10\}$范围内的流量分布任意值, 有唯一缓冲区分配取值与之对应。这就是缓冲区分配方案计算的依据。在步骤 3 中遍历数据通路集合即可获得每条通路对应的缓冲区分配方案。仍要说明的是, 对于映射函数的构造, 在这里只是给出一种实例来说明算法的实现和计算过程, 而实际应用中, 可以根据具体的设计需求和资源约束对构造函数重新定义, 并且, 对于复杂的数据流量分布和系统设计, 构造函数的获取本身也是可以进一步深入研究的问题, 此处不再详述。

考虑 $n\times n$ 的 2-D mesh NOC 系统, 其数据通路总和即为所有节点输出数据通路求和。位于角上位置 4 个节点的输出数据通路个数为 2, 4 条边路上的 $n-2$ 个非角落位置节点的输出数据通路个数为 3,

剩余其他节点的输出数据通路个数均为 4, 从而不难得出 $n \times n$ 的 2-D mesh NOC 系统中的数据通路总和为 $4n(n-1)$。对于 mesh 或 torus 结构, 其系统中数据通路的总数与系统规模成线性关系。由于算法的计算过程采用线性数组实现对数据通路的保存与计算, 该算法计算复杂度与系统规模的增加(节点个数、缓冲区容量)成线性关系。因此随着系统规模的不断增大, 该算法将同样能够胜任系统缓冲区分配方案的实现。

另外, 从上述算法执行流程可以看出, 该算法的缓冲区分配方案计算过程仅依赖于相关数据通路上的流量特征分布, 以及整个系统的资源约束等, 而与系统的拓扑结构无关。

3 实验结果与分析

为了验证本文提出的算法, 用 C++ 构建模拟器, 该模拟器基于 NOC 模拟器 Nirgam 平台, 实现 4×4 2-D mesh 网络。实验过程考虑两大类流量模型, 一类是随机流量模型, 另一类是真实 H.264 编码过程映射后提取的流量分布模型。两种流量分布模型分别从实验与实际应用角度对本文提出的算法进行验证, 其中随机流量模型针对均匀流量分布与不同的热点流量特征进行缓冲区算法实现。

3.1 随机流量评测

采用不同的路由策略针对多种随机流量模型对算法进行测试。如表 2 所示为采用 XY 路由策略, 对均匀流量和热点流量分别采用均匀分配缓冲区的方法(UBUF)和使用本文算法实现的定制缓冲区分配方案(CBUF)后获得的实验数据比较。实验中采用的热点流量有 1hotspot、2hotspot、3hospot, 分别代表 NOC 中有 1、2、3 个热点。1hotspot 热点在 node7 处, 2hotspot 的热点位于 node0 和 node3, 3hotspot 热点位于 node0、node3 和 node7。表中括号内数字为缓冲区分配方案最终使用的缓冲区总容量。表中第 2 列给出采用 UBUF 为路由器每个输入端口分配 1 个缓冲区获得的平均 flit 延迟值(网络中共 48 条数据通路), 第 4 列为 CBUF 缓冲区分配方案获得的平均 flit 延迟值。可以看到, 采用 CBUF 获得的平均 flit 延迟值与 UBUF 相当时, 其使用的缓冲区总容量可降低约 50%。

表 2 平均 flit 延迟(周期)比较(XY 路由)

Tab. 2 Flit latency (cycles) comparison (XY routing)

	UBUF(48)	UBUF(96)	UBUF(144)	CBUF
1hotspot-7	2.22213	1.37778	1.22222	1.22222(56)
2hotspot-0&3	4.79542	4.19292	3.67034	3.63927(86)
3hotspot-0&3&7	3.49894	2.75971	2.25272	2.24381(70)

表 3 所示为采用 OE 路由对各种流量情况下实验结果比较, CBUF 方法相对于 UBUF 来说同样节省了显著的缓冲区容量。

表 3 平均 flit 延迟(周期)比较(OE 路由)

Tab. 3 Flit latency (cycles) comparison (OE routing)

	UBUF(48)	UBUF(96)	UBUF(144)	CBUF
1hotspot-7	2.22351	1.37834	1.21545	1.22358(56)
2hotspot-0&3	3.32774	2.28083	2.16338	2.21736(66)
3hotspot-0&3&7	3.16753	2.69353	2.58326	2.57154(70)

3.2 真实流量评测

本文实现了一个真实的 H.264 编码过程系统映射, 如图 3 所示, 分析获得该应用映射后的数据流量分布特征。图中数字标号表示应用映射后的主要数据通路。

如表 4 所示为采用本文提出的缓冲区分配算法用于该应用映射上获得的结果比较。采用两种不同路由策略进行实验, 结果表明采用 CBUF 获得缓冲区分配方案在保证同等性能情况下节省了约 60% 的缓冲区容量, 这对于受限的 NOC 资源是很可观的。

表 4 H.264 应用实例, 平均 flit 延迟(周期)比较

Tab. 4 H. 264 encoding process, flit latency (cycles) comparison

	UBUF(48)	UBUF(96)	UBUF(144)	CBUF
XY	2. 39966	1. 52001	1. 40000	1. 40000(60)
OE	2. 47946	1. 52	1. 4	1. 38422(58)

上述实验表明, 文中给出的缓冲区分配算法在不同的流量分布模型下, 在不同的路由策略下, 相对于均匀分配缓冲区的方法, 在保证同等性能的情况下都能够获得相应的资源节省。

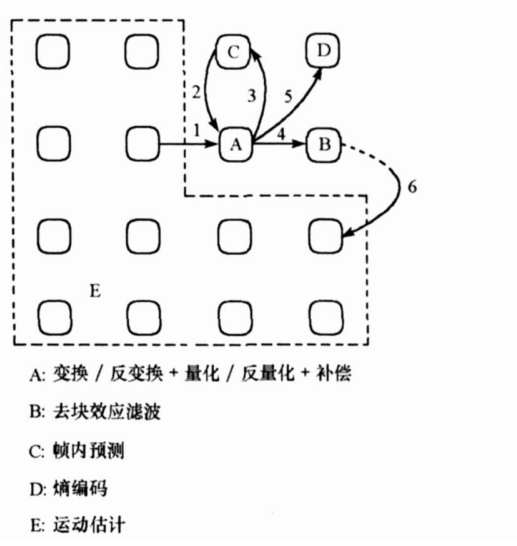

图 3 H. 264 编码过程应用映射

Fig. 3 Application mapping of H. 264 encoding process

4 结论与展望

对于资源受限的 NOC 系统, 为了减小系统实现代价并优化系统性能, 每个端口的缓冲区容量需要面向应用流量特征进行精心分配。本文给出一种面向应用数据负载的缓冲区分配算法, 实现了 NOC 系统中不同路由器端口缓冲区容量的自动分配。在满足系统性能约束的前提下, 最小化缓冲区总容量。实验结果表明在保证同等性能情况下减少 50% ~ 60% 的缓冲区总容量。

采用 2-D mesh 网络作为系统实验平台实例。通过算法计算过程可以看出, 该算法可应用于任意拓扑结构的网络模型中, 其本质是与网络拓扑无关的。本文针对 XY 和 OE 路由策略进行实验, 算法对于其他确定性路由策略同样适用。当前的流量特征采用应用映射分析统计的方法获得, 后续研究工作将针对流量特征的提取建模展开。同时, 为获得更多 NOC 系统相关评测指标, 如功耗等, 可有针对性地加入系统模型或采用 FPGA 进行实现。

参 考 文 献:

[1] Benini L, DeMicheli G. Networks on Chips: A New SoC Paradigm[J]. Computer, 2002, 35(1): 70- 78.

[2] Dally W J, Towles B. Route Packets, Not Wires: On-chip Interconnection Networks[C]// Proc. DAC, 2001: 684- 689.

[3] Hemani A, Jantsch A, Kumar S, et al. Network on a Chip: An Architecture for Billion Transistor Era[C]// Proc. IEEE NorChip Conf., 2000: 166- 173.

[4] Kumar S, Jantsch A, Millberg M, et al. A Network on Chip Architecture and Design Methodology[C]// Proc. Symp. VLSI, 2002: 105- 112.

[5] Saastamoinen I, Alho J N M. Buffer Implementation for Proteo Networks-on-chip[C]// Proc. Int. Symp. Circuits and Syst., 2003: 113- 116.

[6] Liang J, Laffely A, Srinivasan S, et al. An Architecture and Compiler for Scalable On-chip Communication[J]. IEEE Trans. Very Large Scale Integr(VLSI) Syst., 2004, 12(7): 711- 726.

[7] Jalabert A, Murali S, Benini L, et al. xPipesCompiler: A Tool for Instantiating Application-specfic NoCs[C]// Proc. DATE Conf., 2004: 884- 889.

[8] Ogras U Y, Marculescu R. It's a Small World After All: NoC Performance Optimization Via Long Link Insertion[J]. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. (Special Section on Hardware/Software Codesign and System Synthesis), 2006, 14(7): 693- 706.

[9] Hu J, Marculescu R. Energy and Performance-aware Mapping for Regular NoC Architectures[J]. IEEE Trans. Comput. Aided Des. Integrated Circuits Syst., 2005, 24(4): 551- 562.

[10] Mu S, De Micheli G. Bandwidth-constrained Mapping of Cores onto NoC Architectures[C]// Proc. DATE Conf., 2004: 896- 901.

[11] Chiu G. The Odd-even Turn Model for Adaptive Routing[J]. IEEE Trans. Parallel Distrib. Syst., 2000, 11(7): 729- 738.

[12] Glass C J, Ni L M. The Turn Model for Adaptive Routing[C]// 25 Years ISCA: Retrospectives and Reprint, 1998: 441- 450.

[13] Dielissen J, Radulescu A, Goossens K, et al. Concepts and Implementation of the Philips Network-on-chip[C]// Proc. IP-Based SoC Design, 2003.

[14] Huang T C, et. al. Virtual Channels Planning for Networks-on-chip. [C]// Proc. of the ISQED Conference, 2007.

[15] Lai M C, Wang Z Y, Gao L, et al. A Dynamically-allocated Virtual Channel Architecture with Congestion Awareness for On-chip Routers[C]// Proc. of the DAC Conference, 2008.

[16] Hu J, Ogras U Y, Marculescu R. System Level Buffer Allocation for Application-specific Networks-on-chip Router Design[J]. IEEE Trans. on Computer-aided Design of Integrated Circuits and Systems, 2006, 25(3): 2919- 2933.

[17] 王力纬, 曹阳, 李晓辉, 等. 虫孔路由 NOC 的缓冲区分配算法[J]. 北京邮电大学学报, 2008, 31(4).