基于 FPGA 的片上网络自适应路由算法的设计与实现

物联网工程　　12073232　　江岑倩

指导教师　　蔡旻

摘要

随着半导体工艺技术的迅速发展和芯片制造水平的提高，基于总线互连的多核片上系统（System-on-Chip，SoC）出现了许多瓶颈问题。片上网络（Network-on-Chip，NoC）基于计算机网络通信的思想，核与核之间通过分组路由的方法进行通信，克服了这些瓶颈问题。

本文基于蚁群优化（Ant Colony Optimization, ACO）思想，采用适合于 FPGA 实现的信息素表的更新方法和蚂蚁包的路由选择方法，来设计应用于 NoC 的自适应路由选择算法。本文最后将上述路由选择算法与其它的 NoC 路由选择算法进行性能上的比较和分析。实验结果表明，相较于 Buffer Level 选择算法，文中实现的 ACO 选择算法在 Uniform、Transpose、Hotspot 三种输入模式下能达到的吞吐率最大提升幅度为+23.81%（PIR=0.5）、+16.69%（PIR=0.5）和-0.96%（PIR=0.01），能达到的平均包时延最大降低幅度为+3.90%（PIR=0.2）、+9.73%（PIR=0.01）和+7.18%（PIR=0.6）。

关键词：片上网络；自适应路由；蚁群优化；选择算法；现场可编程门阵列

Abstract

With the rapid development of semiconductor processes and chip manufacturing technologies, Bus-based multicore SoCs interconnects encountered performance and energy bottlenecks. NoCs was proposed to implement on-chip interconnects that are inspired by traditional computer networks to enable point-to-point transmission of network packets.

This work focuses on the design and FPGA implementation of ant colony optimization (ACO) based selection algorithms for adaptive routing in NoCs, which consists of pheromone updating and pheromone table based routing. The aforementioned algorithm is then implemented in SystemVerilog and verified by test benches. FPGA Simulation results show that, the aforementioned ACO based selection algorithm can achieve throughput improvement of +23.81% (PIR=0.5), +16.69% (PIR=0.5) and -0.96% (PIR=0.01), and average packet delay reduction of +3.90% (PIR=0.2)、+9.73% (PIR=0.01) 和+7.18% (PIR=0.6), under Uniform, Transpose and Hotspot traffics, as compared to Buffer Level selection algorithm. This work provides great insights on the real-world applications of ant colony optimization and FPGA based implementation of ACO based selection algorithms for adaptive routing on NoCs.

Keywords：network on chip, adaptive routing, ant colony optimization, selection strategy, field programmable gate array

1．绪论

1.1 课题背景

集成电路技术在过去的几十年得到了飞速的发展，在单一芯片上可集成的核心数目也越来越多，随着人们对芯片处理能力要求的不断提高，基于传统的总线互连的多核片上系统逐渐不能胜任处理器核之间的通信需求，于是，有人提出了将计算机网络的实现方式运用到芯片的网络设计中，利用网络将芯片上的各个处理器连接在一起，这就形成了片上网络。目前对 NoC 的研究也很广泛，路由算法就是当前片上网络研究的热点之一。数据密集型应用表现出的对片网络带宽的极大需求，易造成所在结点周边区域的通信拥堵。而自适应路由算法能通过分析网络状态避开这些拥堵的区域，比确定性路由算法更能适应不均匀的片上网络流量分布。

1.2 相关工作

自适应路由算法一般可分为两部分。第一部分为路由算法部分，在路由器转发消息包时，根据路由规则计算出所有可用的输出端口列表，奇偶转弯模型（Odd-Even Turn Model）路由[2]算法就是路由算法部分的一个例子。第二部分为选择算法部分，该部分能通过分析网络状态信息，在上一部分已选出

的可用输出端口列表中选出其认为最优的一个，目前已提出的选择算法包括 Buffer Level（BL，缓冲水平）选择算法、Neighbor on Path（NoP，路径邻居）选择算法等。这两种算法在转发消息包时，考虑的都是当前邻居结点的输入输出缓冲状态，即只考虑当前的网络状态信息，而我们选择设计的基于蚁群优化的选择算法考虑了网络的历史状态。

蚁群优化思想的来源是自然界中的蚂蚁觅食行为。蚁群优化思想借鉴了蚂蚁间基于信息素的信息交流，其呈现出的并行与分布的特性使它能适用于片上网络路由选择算法的设计。国外已有人提出了基于蚁群优化思想的自适应路由算法，如本文实现的选择算法所基于的 AntNet 算法[3][4]就是其中一种。

2．基于蚁群优化的自适应路由选择算法的设计与实现

基于蚁群优化的选择算法相较于其它路由选择算法有几大不同之处，一是多了一种探测网络状态的消息包类型——蚂蚁包，二是基于信息素的路由表的设计，消息包基于信息素表来选择合适的下一跳，三是信息素表只由蚂蚁包来更新。AntNet 算法的基本思想结合了基于信息素表的设计以及基于缓冲水平的本地模型。该算法中的选择概率计算如公式（1）所示：

$$P'(j,d)=\frac{P(j,d)+\alpha L_i}{1+\alpha(|N_k|\text{-}1)} \tag{1}$$

其中 P(j,d)为信息素表中(j,d)位置的信息素浓度值，Li 代表邻居结点的输入输出缓冲水平，N_k 是当前结点的邻居结点数，α 为路由选择因子，它的值代表了邻居结点缓冲水平在选择考虑时所占的比重，P’(j,d)是路由选择依据的概率，即发往目的地结点 d 的消息包、会从 j 输出端口输出概率。另外，该算法中对于信息素浓度的更新计算公式如下：

$P(\mathrm{i})=P(i)+r\times(1-P(i))$，结点在蚂蚁包的路径中 （2）

$P(\mathrm{i})=P(i)\text{-}r\times P(i)$，结点不在蚂蚁包的路径中 （3）

当需要增加信息素浓度时，采用公式（2），当需要减少信息素浓度时，则采用公式（3）。公式中信息素浓度 P(i)的取值范围为 0 到 1，r 为激励因子，代表了信息素浓度增减的幅度。

本文的选择算法设计基于 FPGA 硬件，所以在设计实现时相较于 AntNet 算法有以下几个不同之处：

1）新增两种消息包类型：正向蚂蚁包和逆向蚂蚁包；

2）采用了固定大小的基于信息素的路由表结构；

3）将公式（1）直接作为选择公式，即不进行概率选择，而是直接选择信息素值较大的输出端口；

4）将公式（1）中的选择因子α 设为 0，即暂不考虑邻居结点的缓冲水平；

5）修改公式（2）和（3）中的激励因子 r 的定义，并将其值设为 1。修改后的更新公式如下所示：

$$P(\mathrm{i})=P(i)+\mathrm{r} \tag{4}$$

$$P(\mathrm{i})=P(i)\text{-}\mathrm{r} \tag{5}$$

普通消息包在网络中载有有效载荷，蚂蚁包都不载有有效载荷。正向蚂蚁包和普通消息包都根据信息素表进行路由，而正向蚂蚁包在到达目的地结点后会转换为逆向蚂蚁包，并按照正向蚂蚁包选择的路径原路返回。在逆向蚂蚁包原路返回时，就会更新其所经结点的信息素表中的信息素浓度。

表 1 信息素表设计结构

Destination\Neighbor	北	东	南	西
0				
1				
:				
d	ph[d][北]	ph[d][东]	ph[d][南]	ph[d][西]
:				
结点数-1				

本文中的信息素表结构的设计如表 1 所示，它有着固定的行数和列数，其中行数等于网络中的结点数，列数等于该结点的除了本地 PE（计算单元）以外的输入输出端口数。

假设当前路由器结点要转发一个目的地为 d 结点的正向蚂蚁包，而在路由算法部分已经选出了两个可用的输出端口，北输出端口和西输出端口。然后，本设计中的选择算法将在信息素表中找出这两个输出端口对应的信息素浓度值 ph[d][北]、ph[d][西]，判断出信息素浓度值较大的那个端口，并将其选为最终的输出端口。

假设当前路由器结点从某一个输入端口（如北）接收到一个原目的地为 d 结点的逆向蚂蚁包，则会根据更新算法增加信息素表中对应的 d 行的 ph[d][北]值，并减少表中 d 行的其它信息素浓度值。

3．性能测试

3.1 性能指标

本实验文对片上网络的吞吐率（Throughput）和平均包时延（Average Packet Delay）这两种性能指标进行了测试，本文将分别对其进行分析。吞吐率的计算公式如公式（6）所示，平均包延时的吞吐率的计算公式如公式（7）所示。

$$\text{吞吐率} = \frac{\text{测试周期内结点成功发送的总的消息包数目}}{\text{总的测试周期数} \times \text{网络中总的结点数}} \tag{6}$$

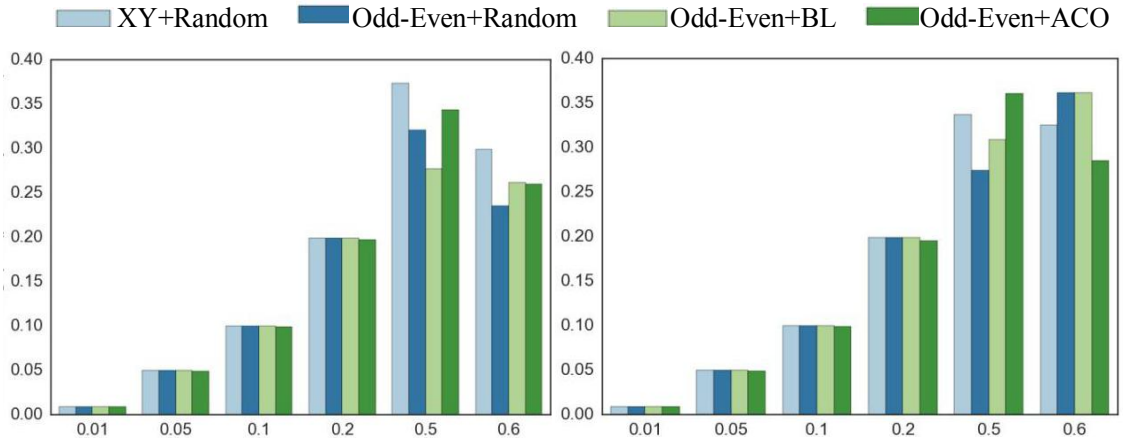
$$\text{平均消息包时延} = \frac{\sum \text{每个消息包的在网络中花费的周期数}}{\text{总的消息包数目}} \tag{7}$$

3.2 实验结果与分析

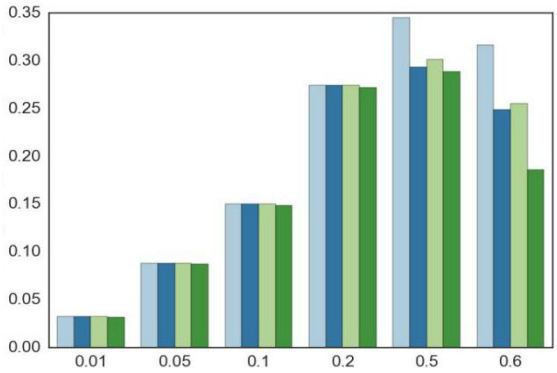
本实验使用 Altera Quartus Prime 15.1 开发环境和 SystemVerilog 硬件描述语言，采用了 4x4 二维网格（2D-Mesh）拓扑结构和 4 个包长度的缓冲空间，模拟了三种 Traffic（Uniform，Transpose 和 Hotspot）状态，选用了 0.01、0.05、0.1、0.2、0.5、0.6 这六种消息包注入率（Packet Injection Rate，PIR）分别 d 对吞吐率和平均包时延进行测试，测试周期数为 10000 cycle.

实验在路由算法部分选用并实现了 X-Y 路由和奇偶（Odd-Even）转弯模型路由，在选择算法部分实现了随机（Random）路由选择、缓冲水平（BL）路由选择和基于蚁群优化（ACO）的路由选择，odd even 与三种选择算法进行组合可形成三种自适应路由，本实验对 XY+Random、odd even+Random、odd even+BL、odd even+ACO 这四种组合的路由算法分别进行了测试，并重点分析 odd even+ACO 路由算法所表现的性能与其它路由算法所表现的性能的差异。

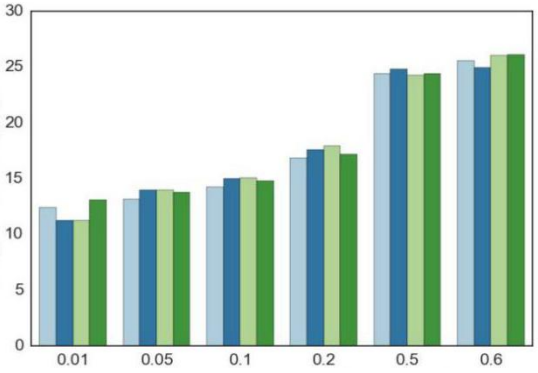
实验结果如图 1 所示。图 1（1）-1（3）分别给出了 XY+Random、Odd-Even+Random、Odd-Even+BL、Odd-Even+ACO 四种组合的路由算法在 Uniform、Transpose、Hotspot 三种输入下的吞吐率。图 1（4）-1（6）分别给出了这四种组合的路由算法在 Uniform、Transpose、Hotspot 三种 traffic 输入下的平均包时延。



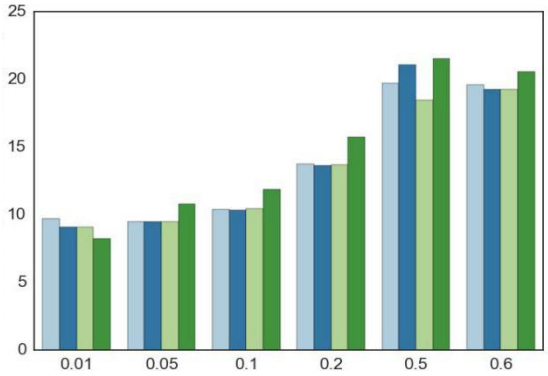
（1）Uniform 输入下的吞吐率　　（2）Transpose 输入下的吞吐率



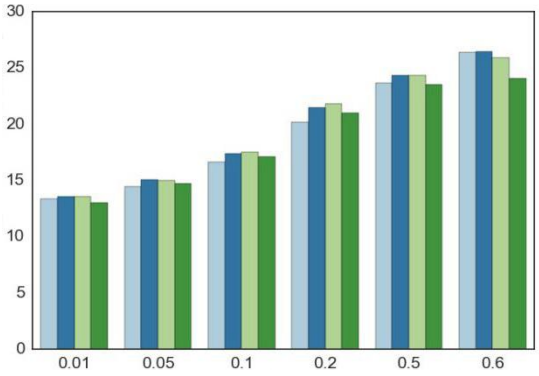
（3）Hotspot 输入下的吞吐率



（4）Uniform 输入下的平均包时延



（5）Transpose 输入下的平均包时延



（6）Hotspot 输入下的平均包时延

图 1　NoC 吞吐率和平均包时延性能

从图中可以看出，在消息包注入率较低时，4 种算法的吞吐率与注入率是平衡的。而当注入率较高时，Odd-Even+ACO 算法的吞吐率性能表现时好时坏。对于平均包时延性能，在 Uniform 输入下，4 种算法的性能表现差别不大，在 Transpose 输入下，Odd-Even+ACO 算法的性能表现会差一些，而在 Hotspot 输入下，Odd-Even+ACO 算法的性能表现会好一些。

4．结论

本文设计并实现了一种适用于 NoC 的基于蚁群优化思想的自适应路由选择算法。经过性能分析以及与其它算法的比较后发现，本文中实现的选择算法的性能表现不会太高也不会太差。在设计实现过程中，总结出了以下可改进的地方：完善 FPGA 中实现的 NoC 架构，在进行路由选择时加入网络当前状态的考虑，实现根据信息素表浓度值地概率选择，构建更完善的 Testbench 平台，使用更多结点数进行测试，测试和发现蚂蚁包的更佳注入率，以及其它参数的设置等。在以后的工作中会继续考虑和研究这些可改进之处。

参考文献

[1] Enright, N. Peh, L. On-Chip Networks. In Synthesis Lectures on Computer Architecture. Morgan & Claypool, 2009.

[2] G.-M. Chiu. The Odd-Even Turn Model for Adaptive Routing. IEEE Trans. Parallel Distrib. Syst. vol. 11, no. 7, pp. 729-738, July 2000.

[3] Di Caro G., Dorigo M. AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research. 9, pp. 317-365, 1998.

[4] M. Daneshtalab and A. Sobhani. NoC Hot Spot Minimization Using AntNet Dynamic Routing Algorithm. In Proc. IEEE Appl. Specific Syst., Architect. Processors Conf. 2006, pp. 33-38.