计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150682
53(4):921-931,2016

二次改进遗传算法与3D NoC低功耗映射

张大坤　宋国治　林华洲　任淑霞
(天津工业大学计算机科学与软件学院　天津　300387)
(Zhandakun2013@163.com)

Double Improved Genetic Algorithm and Low Power Task Mapping in 3D Networks-on-Chip

Zhang Dakun, Song Guozhi, Lin Huazhou, and Ren Shuxia
(*School of Computer Science & Software Engineering, Tianjin Polytechnic University, Tianjin* 300387)

Abstract　With the rapid development of integrated circuit technology, the number of integrated components on a chip continues to increase. Efficient interconnection between the processing units on chip becomes a key issue. Therefore firstly system-on-chip (SoC) and then two-dimensional networks-on-chip (2D NoC) have been proposed to cope with this problem. But now even the 2D NoC has reached a bottleneck in many aspects, so the design of Three-Dimensional networks-on-chip (3D NoC) is inevitable. 3D NoC has attracted the attention of the researchers from both Academia and industry. One of the key issues of 3D NoC is low-power mapping. We have previously proposed a 3D NoC low-power mapping algorithm based on improved genetic algorithm with good results. But when the scale of the problem gets larger, the amount of calculation increases gradually and operation efficiency is reduced significantly. To solve this problem, this paper proposes a new 3D NoC task mapping algorithm with power optimization based on a double improved genetic algorithm, and the simulation experiments are conducted to validate the algorithm. The results show that under the conditions of a large population size, the 3D NoC task mapping algorithm cannot only reduce the power, but also reduce the running time significantly.

Key words　3D network-on-chip (3D NoC); low-power mapping; improved genetic algorithm; double improved genetic algorithm; Greedy algorithm

摘　要　随着集成电路技术的迅速发展,芯片的集成度不断提高,片上众多处理单元间的高效互连成为关键问题,因而相继出现了片上系统(system-on-chip, SoC)和二维片上网络(two-dimensional network-on-chip, 2D NoC).当二维片上网络在多方面达到瓶颈时,三维片上网络(three-dimensional network-on-chip, 3D NoC)应运而生.三维片上网络已引起学术界和产业界的高度重视,三维片上网络低功耗映射是其中的1个关键问题.之前的研究曾提出过一种基于改进遗传算法的3D NoC低功耗映射算法,并收到了良好的仿真效果.但当问题规模变大时,计算量随之增大、运行效率明显降低.针对这一问题,对3D NoC中面向功耗优化的二次改进遗传算法任务映射机制进行研究,提出了一种新的3D NoC低功耗映射算法,并对该映射算法进行了仿真实验.实验结果表明,在种群规模较大的条件下,该算法不仅能够继续降低功耗,而且能够大幅度地减少映射算法的运行时间.

关键词　三维片上网络;低功耗映射;改进遗传算法;二次改进遗传算法;贪心算法

中图法分类号　TP391

收稿日期:2015-07-23;**修回日期**:2016-01-22

基金项目:国家自然科学基金项目(61272006)

This work was supported by the National Natural Science Foundation of China (61272006).

1958年世界诞生了第1个集成电路,随着集成电路规模的不断增大,片上系统(system-on-chip, SoC)应运而生,而且SoC的处理单元(processing elements, PE)数不断增加,为高效地连接数量巨大的PE,产生了二维片上网络(two-dimensional network-on-chip, 2D NoC)这种主流的片上互连架构.而2D NoC在面积、功耗、布局布线以及封装密度等方面都已达到了瓶颈,三维片上网络(three-dimensional network-on-chip, 3D NoC)应运而生.自2005年第1篇以3D NoC为主题的学术论文发表以来[1],3D NoC以其更短的全局互连、更高的性能、更低的互连损耗、更高的封装密度以及更小的体积等诸多优势成为1个重要的研究方向[2-3].降低功耗是3D NoC所面临的1个关键问题,从多种途径降低3D NoC的功耗非常必要.映射决定知识产权(intellectual property, IP)核在3D NoC上的位置,通过改进映射算法能够有效地降低3D NoC的功耗.因此,3D NoC低功耗映射已逐渐成为3D NoC领域的研究热点.3D NoC映射问题和任务调度问题相似,都是NP难解问题[4].有许多研究者尝试应用各种算法解决3D NoC映射问题,其中,美国宾夕法尼亚州立大学Charles[5]采用遗传算法研究了考虑到热感知和通信感知的3D NoC映射问题;南京大学王佳文[6-8]、汤普森河大学的Elmiligi等人[9]、航空电子系统综合技术重点实验室Ge等人[10]均尝试过采用遗传算法解决3D NoC映射问题,并取得了相应的研究成果.本文将对3D NoC中面向功耗优化的二次改进遗传算法任务映射机制进行研究.

1 3D NoC低功耗映射问题

1.1 基本概念

3D NoC低功耗映射就是在给定通信轨迹图和拓扑结构图的基础上,研究如何将IP核映射到3D NoC的资源节点上,使得整个3D NoC的通信量最小,达到降低通信功耗的目的.为了描述3D NoC映射问题,给出如下定义:

定义1. 通信轨迹图 $CTG(N,C)$.设 $CTG(N,C)$为有向图,其中 N 为顶点集,节点 $n_i \in N$ 表示1个IP核;C 为边集,有向边 $c_{ij} \in C$ 表示节点 c_i 到节点 c_j 的边,w_{ij} 表示边 c_{ij} 的权重.

定义2. 拓扑结构图 $TAG(T,E)$.设 $TAG(T,E)$为有向图,其中,T 为PE集合,$t_i \in T$ 表示1个PE;E 为边集,有向边 e_{ij} 表示节点 t_i 到节点 t_j 的边[11-12].

1.2 3D NoC映射算法评估标准

目前尚没有一致公认的面向3D NoC映射问题的评估标准.3D NoC作为解决片上系统通信问题的解决方案.其中,网络吞吐量与平均延迟都是重要的性能指标;3D NoC映射算法的设计也应当考虑功耗问题,过高的功耗可能会对网络产生致命的影响;3D NoC的三维立体结构,会引发网络内部发热不均的问题,发热过高可能会导致整个网络的瘫痪;对于实时性要求相对高的应用,最大网络延时也应作为网络性能的重要指标.文献[13-14]提出了评价映射算法优劣的4个重要性能指标:

1) 平均网络延迟.3D NoC的出现主要是为了解决SoC内资源节点之间的通信问题,因此,平均网络延迟是衡量3D NoC设计质量的重要指标,平均网络延迟间接地决定了3D NoC的吞吐率.

2) 最大网络延迟.目前大多数针对3D NoC的延时模型都是平均延时模型,然而在出现拥塞的状况下,决定整个网络通信效率的往往是最大延迟,当最大延迟超过了其容错上限,将导致整个网络无法正常工作.因此,最大延迟也是3D NoC评估的重要指标.

3) 平均功耗.3D NoC在传递消息的过程中会发生能量的消耗,平均功耗能够反映3D NoC在传送数据时功耗性能.

4) 发热与负载均衡.3D NoC由于其结构特性,内部节点容易出现发热过高的情况,过热点的产生将影响整个芯片的性能.过热点区域位于网络边缘容易散热,可以降低过热点区域对整个网络性能的影响.因此,在网络设计时尽可能地将过热点区域分散到网络边缘.

1.3 3D NoC功耗模型

1.3.1 3D NoC功耗计算公式

3D NoC中的功耗主要来自2方面:路由节点上的功耗和链路上的功耗.路由节点上的功耗主要是路由节点内部存储、交换、路由内部连线和路由选择所产生的能耗;链路上的功耗指的是路由数据经过路由节点之间链路时所产生的能耗.文献[11]给出了3D NoC的功耗模型,节点 t_i 发送1个微片(flit)到节点 t_j 消耗的能量表示为

$$E_{\text{bit}}^{t_i,t_j} = \mu E_{R\text{bit}} + \mu_{\text{H}} E_{L_{\text{H}}\text{bit}} + \mu_{\text{V}} E_{L_{\text{V}}\text{bit}}, \quad (1)$$

其中,μ 表示节点 t_i 到节点 t_j 经过的路由器个数,μ_{H} 和 μ_{V} 分别是节点 t_i 到节点 t_j 经过的水平方向和垂直方向的边数(跳步),$E_{R\text{bit}}$ 表示1个微片通过1个路由器时所消耗的能量,$E_{L_{\text{H}}\text{bit}}$ 和 $E_{L_{\text{V}}\text{bit}}$ 分别表示

1 个微片通过 1 条水平方向和垂直方向的线路时的耗能. 文献[14]给出的 $E_{L_{\mathrm{H}}\mathrm{bit}}$ 和 $E_{L_{\mathrm{V}}\mathrm{bit}}$ 的计算公式为

$$E_{L_{\mathrm{H}}\mathrm{bit}}=d_{\mathrm{H}}V_{dd}^2C_{wire_{\mathrm{H}}}/2, \quad (2)$$

$$E_{L_{\mathrm{V}}\mathrm{bit}}=d_{\mathrm{V}}V_{dd}^2C_{wire_{\mathrm{V}}}/2, \quad (3)$$

其中,d_{H} 和 d_{V} 分别表示水平方向和垂直方向线路的长度,V_{dd} 表示供电电压,$C_{wire_{\mathrm{H}}}$ 和 $C_{wire_{\mathrm{V}}}$ 分别表示水平方向和垂直方向线路的电容.

1.3.2 映射目标函数

3D NoC 映射就是寻找 IP 核与 PE 的 1 个对应关系. 在给定通信轨迹图 CTG 和拓扑结构图 TAG 的条件下寻找 1 个映射函数 $M(\cdot):N\rightarrow T$,使得总功耗最低(用 $E_{\min}$ 表示),其目标函数如下:

$$E_{\min}=\min\Big(\sum_{n_i,n_j\in N}(w_{ij}\times E_{\mathrm{bit}}^{M(n_i),M(n_j)})\Big), \quad (4)$$

且满足条件:

$$\forall\, n_i\in N, M(n_i)\in T, \quad (5)$$

$$\forall\, n_i\neq n_j, M(n_i)\neq M(n_j). \quad (6)$$

对于确定的拓扑结构,$E_{R\mathrm{bit}}$,$E_{L_{\mathrm{H}}\mathrm{bit}}$ 和 $E_{L_{\mathrm{V}}\mathrm{bit}}$ 是确定的. 由文献[11]可知,线路上的功耗与线路的长度成正比,所以,2 个通信节点之间的线路长度是影响 3D NoC 功耗的关键要素. 由于硅穿孔(through silicon via, TSV)技术的出现,3D NoC 在垂直方向的线路长度远远小于水平方向的长度,因此,传输同样的数据量,在垂直方向的功耗远远小于水平方向的功耗.

2 基于改进遗传算法的 3D NoC 低功耗映射

本文作者曾对 3D NoC 映射问题进行了研究,提出一种基于改进遗传算法(improved genetic algorithm, IGA)的 3D NoC 低功耗映射算法(简称 IG 映射算法)[15],现对其进行简单回顾.

2.1 问题简介

如引言所述,用遗传算法解决 3D NoC 映射问题已取得一些研究成果[5-11],从映射出发降低 3D NoC 功耗是 1 个有效的途径. 遗传算法根据问题的目标函数构造 1 个适应值函数,产生由多个解构成的初始种群,根据适应值的优劣选择种群中的个体进行遗传操作产生新的种群,当进化达到终止条件后,选择适应值好的个体作为问题的解. 算法的实现主要分为如下 5 个步骤:

算法 1. 遗传算法.

Step1. 遗传算法在运行之前要将解空间的数据表示成算法运行时可用的基因数据,由于每个 IP 核只能映射到 1 个 PE 上,因此,在映射问题中采用整数编码.

Step2. 随机产生 1 个初始种群,种群中每个个体都对应映射算法的 1 个解.

Step3. 根据问题的目标函数构造适应值函数,计算种群中每个个体的适应值.

Step4. 根据适应值的优劣不断地进行选择和繁殖操作,以得到下一代种群.

Step5. 达到终止条件后,选择适应值最好的个体作为算法的最终解.

基本遗传算法的初始种群是随机生成的,这使得初始种群存在个体分布不均匀、初始种群质量偏低的问题. 作者提出的 IG 映射算法是将贪心算法与遗传算法相结合,用以解决 3D NoC 低功耗映射问题,相对于传统遗传算法具有更优的搜索能力.

2.2 IG 映射算法的核心思想

IG 映射算法是贪心算法与遗传算法的组合,即采用贪心算法生成初始种群的遗传算法,贪心算法思想主要体现在生成初始种群个体上. 用贪心算法生成初始种群个体主要分为 3 个步骤:

算法 2. 贪心算法.

Step1. 随机地将任务图中的 1 个节点(即 IP 核)映射到 3D NoC 的第 1 个位置(把所有 IP 核从 1 到 N 进行编号,N 为任务图上 IP 核的个数,n 代表任意 1 个 IP 核;把所有 PE 从 1 到 M 进行编号,M 为 3D NoC 上 PE 的个数,m 代表任意 1 个 PE).

Step2. 分别计算将所有 IP 核集合中可用数字放入所有 PE 集可用位置后的适应值,用$\langle n,m,fit\rangle$表示未使用数字 n 放入 3D NoC 的第 m 个位置使得适应值 fit 最大,将$\langle n,m,fit\rangle$放入集合 F 中,从 F 中选择 fit 最大的 3 组,随机地从这 3 组中选择一组$\langle n,m,fit\rangle$,将 n 放到 3D NoC 的第 m 个位置,把 n 从可用的 IP 核集合中删除,把 m 从 3D NoC 的可用位置集合中删除.

Step3. 重复 Step2,直到没有未使用的数字(IP 核集合为空),即任务图中的所有节点都已映射到 3D NoC 上.

生成初始种群后,采用遗传算法进行遗传操作,得到最优个体,然后用最优个体生成仿真器能够识别的通信模式文件,仿真器读取通信模式文件进行仿真得到映射功耗. 仿真结果表明,IG 映射算法可以有效地降低功耗,从总体趋势上看,随着处理单元数目的增加,功耗降低的幅度逐渐增大,在种群规模为200、处理单元为120的情况下,总功耗可降低 14.42%[15].

3 二次改进遗传算法与 3D NoC 低功耗映射研究

3.1 问题的提出与算法描述

3.1.1 问题的提出

2.2 节中的 IG 映射算法是采用贪心算法生成初始种群中的每个个体,使初始种群的质量整体得到了提升,有效地降低了功耗.但该算法在生成初始种群时,单个个体生成的计算量较大,在确定 1 个 IP 核映射到 3D NoC 上时,会计算所有未映射到 3D NoC 上的 IP 核放到 3D NoC 上所有可用位的适应值,而且生成每个个体都要进行同样的运算,所以该算法在生成初始种群时运行效率明显低于基本遗传算法.针对这一问题,对 3D NoC 中面向功耗优化的二次改进遗传算法任务映射机制进行研究,提出了一种基于改进贪心算法和遗传算法的 3D NoC 低功耗映射策略,并称其为基于二次改进遗传算法(double improved genetic algorithm, I²G)的 3D NoC 低功耗映射算法并称之为 I²G 映射算法.

3.1.2 I²G 映射算法的核心思想

IG 映射算法是用一般的贪心算法生成初始种群,而 I²G 映射算法则采用改进贪心算法生成初始种群.I²G 映射算法中采用的改进贪心算法与 IG 映射算法中采用的贪心算法的区别在于:

1) 2.2 节中贪心算法在 Step1,随机地将 1 个 IP 核(用 n 表示)映射到 3D NoC 的第 1 个位置(数字 n 可重复出现),而改进贪心算法中在第 1 个位置上放置的 IP 核(n)不可重复出现($n=1,2,\cdots,N$).

2) 2.2 节中贪心算法在 Step2,是从集合 F 中找出 3 组 fit 最大的〈n,m,fit〉,随机地从中选 1 组,而改进贪心算法只选取 fit 值最大的 1 组〈n,m,fit〉,将 n 放到 3D NoC 的第 m 个位置,把 n 从可用的 IP 核集合中删除,把 m 从 3D NoC 的可用位置集合中删除;重复 2)直到 IP 核集合为空,即表示生成了 1 个个体.

3) 2.2 节中贪心算法可以生成任意多个个体,而改进贪心算法只能生成 N 个个体(N 为任务图规模).因此,改进贪心算法会对所有已生成的个体进行变异操作生成一定数量的邻居个体,最后从所有生成的个体中选取一定数量的个体作为初始种群.

I²G 映射算法的具体实现详见 3.2.3 节.

3.2 种群个体向量表示与初始种群生成

3.2.1 种群个体向量表示

由于每个 PE 都可以向其他任意 1 个 PE 发送数据,CTG 中的 IP 核可以映射到任意 1 个可用的 PE 上.因此,在 3D NoC 映射中种群个体采用整数编码,假设个体 $X=(x_1,x_2,\cdots,x_n)$,n 为 CTG 的 IP 核个数,x_i 为 IP 核编号,个体 X 的编码为 1 到 n 的 1 种排列,通过对编码进行解码即可得到 1 种映射方案.1 个简单的 16 节点视频对象平面解码器芯片(video object plane decoder, VOPD)的通信轨迹图如图 1 所示[16],以此图为例来说明种群个体的向量表示.图 1 中箭头附近的数字表示节点间的通讯量,单位是 MBps.

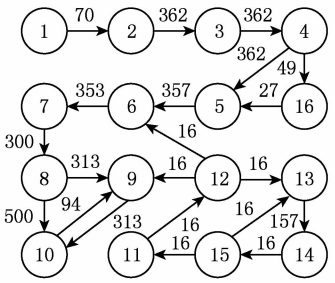

Fig. 1 VOPD communication trace diagram.

图 1 VOPD 通信轨迹图

VOPD 中的 IP 核用数字 1,2,…,16 表示,将 VOPD 映射到 2×3×3 的 3D NoC 上,3D NoC 上的每个 PE 用数字 t_1 到 t_{18} 表示,t_1 到 t_6 表示第 1 层,t_7 到 t_{12} 表示第 2 层,t_{13} 到 t_{18} 表示第 3 层,如图 2 所示.个体 $X=(2,3,4,1,16,5,7,6,11,15,14,12,13,10,8,9)$,表示将 IP 核集(2,3,4,1,16,5,7,6,11,15,14,12,13,10,8,9)分别映射到处理单元 t_1 到 t_{16} 上,映射结果如图 3 所示.将所有 IP 核映射到 3D NoC 上之后,计算 3D NoC 上所有节点之间的通信总量,

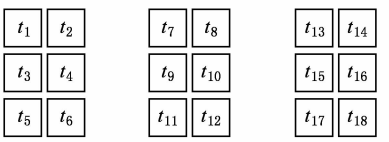

Fig. 2 3D NoC topological structure

图 2 3D NoC 拓扑结构

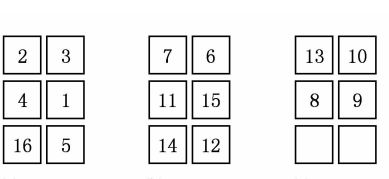

Fig. 3 Mapping result

图 3 映射结果

用遗传算法根据 3D NoC 上的通信总量计算其适应值,通信总量越大,适应值越小,用遗传算法根据适应值大小进行遗传操作.

3.2.2 应用改进贪心算法生成初始种群

应用本文提出的改进贪心算法生成初始种群(Pop)的主要过程如下:

算法 3. 改进贪心算法.

Step1. 将数字 i 放到个体 X 的第 1 个位置($i=1,2,\cdots,N$,当选取每个 i 后,重复 Step2 到 Step5,每次循环都生成 1 个个体).

Step2. 初始化可用集合 $P=\{1,2,\cdots,N\}$,N 为 CTG 的 IP 核个数,将 i 从可用集合 P 中删除.

Step3. 遍历可用集合 P 中的数字 n,将 n 放入个体 X 的所有可用位置,计算放入这些位置后 X 的适应值,从这些适应值中找到最大的适应值 fit,记下 n 放到使得 X 适应值最大的位置 m,把〈n,m,fit〉存到集合 F 中.

Step4. 遍历集合 F,找到 fit 值最大的 1 个元素〈n,m,fit〉,把 n 放到个体 X 的第 m 个位置,并把 n 从可用集合 P 中删除.

Step5. 重复 Step3 和 Step4,直到可用集合 $P=\varnothing$,当 $P=\varnothing$ 时,表示产生了新个体 X,把 X 加入临时种群 $tempPop$ 中.

Step6. 重复上述 Step5,生成 N 个个体.

Step7. 将个体 X 中的任意 2 个坐标数字交换(即变异操作),生成 1 个新个体 Y,即 X 的邻居个体.用此方法生成 20 个 X 的邻居个体放入临时种群 $tempPop$ 中.

Step8. 从临时种群 $tempPop$ 中选取一定规模适应值最大的个体放入 Pop 中.

3.2.3 I^2G 映射算法实现

1) 低功耗映射实现流程

算法 4. I^2G 映射算法.

Step1. 采用改进贪心算法生成初始种群(详见 3.2.2 节).

Step2. 用遗传算法对初始种群进行进化操作,得到算法的最优解.

Step3. 根据最优解生成仿真平台能够识别的通信模式(traffic pattern)文件.

Step4. 仿真平台读取通信模式文件进行仿真,得到 3D NoC 低功耗映射仿真结果.

2) 通信模式文件的生成过程

算法 5. 通信模式文件生成.

Step1. 将 $CTG(N,C)$ 里的边集 C,根据改进遗传算法得到最优个体 X,转换成 TAG 的边集 E.例如:最优个体 $X=(2,3,4,1,8,5,7,6)$,边 c_{12},边 c_{12} 的权重 $w_{12}=100$,映射到 TAG 上后得到边 e_{41},边 e_{41} 的权重 $w_{41}=100$.统计所有边的通信量之和 $wSum$.

Step2. 生成随机数 $1\leqslant r\leqslant n$,如果第 r 条边为 e_{13},则在通信模式文件中写入〈1 3〉,如果通信量等于 0,则循环查找下一条边,直到找到通信量大于 0 的边,在通信模式文件中写入该条边的信息,将该条边的通信量减 1.

Step3. 重复 Step2,直到所有边的通信量为 0.

4 仿真实验与结果分析

4.1 仿真平台与参数设计

4.1.1 仿真平台选择

仿真实验是在 Ubuntu13 操作系统下,采用 C++语言编写算法的实现程序,在 codeblocks 12.11 环境下,采用 Access Noxim 0.2 作为仿真软件进行映射算法的仿真(Access Noxim 0.2 对主流的 2D NoC 仿真软件 noxim 进行了改进,改进后可以用于 3D NoC 的仿真实验).

4.1.2 拓扑结构与路由选择

1) 拓扑结构的选择

3D Mesh 具有结构规则、布线简单、网络节点位置平等且在垂直方向互连采用 TSV 技术减小了总体布线长度等优点,实验中采用了 3D Mesh 结构.

2) 路由选择

XYZ 路由算法易于理解与实现,是主流的路由算法,实验中采用了该路由算法.

4.1.3 参数设置

1) 算法的参数设置.种群规模为 200,遗传迭代次数为 100,交叉率为 0.9,变异率为 0.02.

2) 仿真软件的参数设置.数据包采用无记忆泊松分布(memoryless Poisson distribution)方式注入,数据包的注入率(packet injection rate)设为 0.02;仿真软件统计循环 5 000 次的总功耗;数据包的大小介于 2 个微片到 10 个微片之间;路由器每个通道的缓存大小设为 8 个微片.

4.1.4 硬件运行环境

整个仿真实验的硬件环境是:CPU 为 Intel® Core™ i3-2120,主频为 3.30 GHz,内存为 4 GB 的微型计算机.

4.2 仿真实验结果分析

采用 C++语言编写了基于 IG 映射算法[15]和

基于 I²G 映射算法的 3D NoC 低功耗映射程序，在上述环境下进行了仿真实验.

4.2.1 经典任务图的功耗对比

1) 针对 VOPD 的功耗对比

首先针对 2 个经典的通信轨迹图 VOPD 和 DVOPD(double video object plane decoder)进行仿真实验. 其中，VOPD 中有 16 个节点，如图 1 所示，将 VOPD 映射到 2×2×4 的 3D NoC 上. 仿真实验中，分别采用 IG 映射算法和 I²G 映射算法对该任务图进行映射操作，用得到的最优解生成通信模式文件，仿真器 Access Noxim 通过读取通信模式文件进行映射仿真，得到映射功耗. 用实验数据分别对 2 种映射算法得到的最小功耗、平均功耗和最大功耗进行了比较，如图 4 所示：

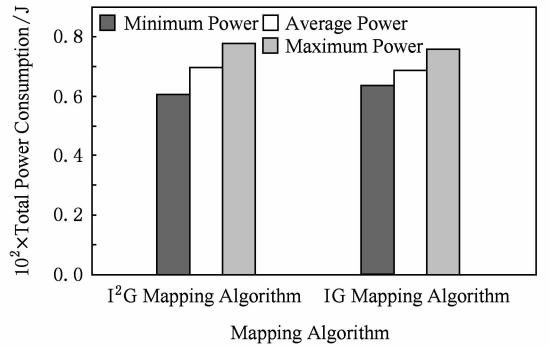

Fig. 4 Comparison of two mapping algorithms' power consumption for VOPD.

图 4 针对 VOPD 的 2 种映射算法功耗对比

分析针对 VOPD 的 2 种映射算法仿真结果可见，在最低功耗方面 I²G 映射算法低于 IG 映射算法，即采用 I²G 映射算法的功耗更低，但是降低的幅度较小；而在平均功耗和最大功耗方面，I²G 映射算法的功耗高于 IG 映射算法的功耗，I²G 映射算法与 IG 映射算法的唯一区别就是生成初始种群时采用的贪心策略不同. 对任务图分别进行实验，由实验结果可知：I²G 映射算法与 IG 映射算法相比，最低功耗降低了 2.02%、平均功耗增加了 2.15%、最高功耗增加了 1.54%. 其中，I²G 映射算法的功耗高于基于 IG 映射算法[16]的功耗，这是由于 VOPD 只有 16 个节点(节点过少)，采用改进贪心策略得到的初始种群容易陷入局部最优.

2) 针对 DVOPD 的功耗对比

通信轨迹图 DVOPD 有 32 个节点，如图 5 所示[16]，图 5 中箭头附近的数字表示节点间的通讯量，单位是 MBps. 实验中将 DVOPD 映射到 2×2×4 的 3D NoC 上，针对通信轨迹图 DVOPD 分别用 2 种映射算法进行了仿真实验.

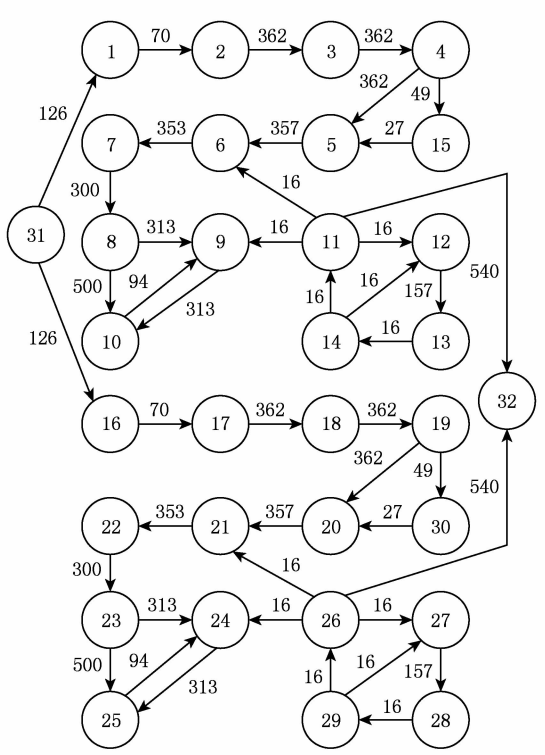

Fig. 5 DVOPD task communication diagram.

图 5 DVOPD 任务通信图

实验中对任务通信图 DVOPD 分别采用 IG 映射算法和 I²G 映射算法的仿真结果如图 6 所示. I²G 映射算法得到的仿真结果明显优于 IG 映射算法的仿真结果：I²G 映射算法与 IG 映射算法相比，最低功耗降低了 13.37%、平均功耗降低了 9.18%、最高功耗降低了 7.48%. 功耗降低显著，这是由于 DVOPD 有 32 个节点，任务规模较大，采用改进贪心策略得到的初始种群不容易陷入局部最优. 由以上实验和分析可见，当任务图规模增大时，I²G 映射算法的优势更为明显.

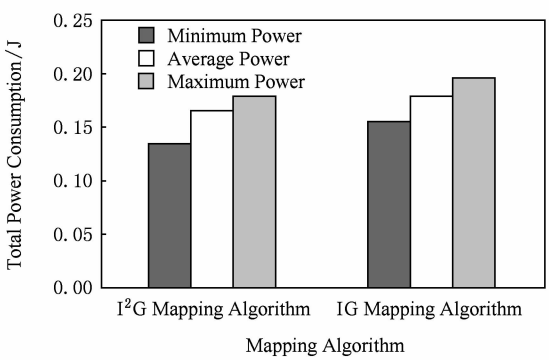

Fig. 6 Comparison of two mapping algorithms' power consumption for DVOPD.

图 6 针对 DVOPD 的 2 种映射算法功耗对比

4.2.2 经典任务图映射算法运行时间对比

IG 映射算法和 I²G 映射算法的运行时间对比如图 7 所示.由图 7 可见,I²G 映射算法的运行时间明显少于 IG 映射算法的运行时间.对于 VOPD,由于任务图规模较小,生成每个个体所用的时间较短,因此 I²G 算法的运行时间减少幅度较小,比 IG 映射算法减少了 19.11%(I²G 映射算法的运行时间为 3.619 s、IG 映射算法的运行时间为 4.474 s);而对于 DVOPD,I²G 映射算法的运行时间比 IG 映射算法的运行时间[15]减少幅度增大、减少了 68.90%(I²G 映射算法的运行时间为 5.444 s、IG 映射算法的运行时间为 17.503 s).

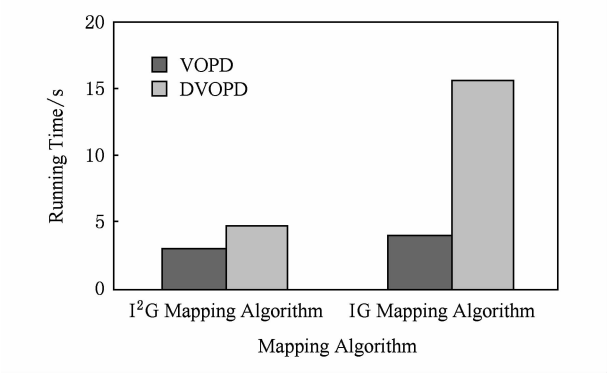

Fig. 7 Comparison of two mapping algorithms' running time.

图 7 2 种映射算法运行时间对比

4.2.3 200 种群规模下随机任务图的功耗对比

1) 仿真实验数据

采用任务生成器 TGFF[17]生成随机任务图,针对不同 IP 核数的 *CTG*、种群规模为 200 时,分别采用 IG 映射算法和 I²G 映射算法求解 10 次,得到的功耗如表 1 和表 2 所示(对仿真实验数据从低到高进行了排序).

Table 1 Experimental Results of Ten Simulation for IG Mapping Algorithm

表 1 IG 映射算法 10 次仿真实验结果

Serial Number	Mapping Power/mJ					
	21 IP Cores	39 IP Cores	59 IP Cores	82 IP Cores	101 IP Cores	121 IP Cores
1	5.2095	9.6546	15.9159	31.5827	40.2432	46.1707
2	5.3288	9.9548	15.9256	32.3326	40.7171	47.4254
3	5.4872	9.9895	16.2326	33.3372	41.0368	49.5953
4	5.7046	10.0015	16.8827	33.6810	41.0506	49.9114
5	5.9681	10.3643	17.0724	33.8693	41.6201	50.5929
6	6.2972	10.3958	17.2947	35.1989	41.9527	51.4930
7	6.7898	11.0041	17.4753	35.4125	42.2520	51.5675
8	6.9874	11.2156	17.6792	36.4021	43.1624	51.5773
9	7.1365	11.2600	17.7615	36.9169	46.3052	52.4319
10	7.7188	11.6521	19.8238	39.3961	47.5627	55.2943

Table 2 Experimental Results of Ten Simulation for I²G Mapping Algorithm

表 2 I²G 映射算法 10 次仿真实验结果

Serial Number	Mapping Power/mJ					
	21 IP Cores	39 IP Cores	59 IP Cores	82 IP Cores	101 IP Cores	121 IP Cores
1	4.2997	9.0698	15.3389	28.5000	36.9035	35.7187
2	4.4969	9.1564	15.4124	28.9428	37.5093	40.4328
3	5.1725	9.1774	15.8190	28.9568	37.8696	40.6130
4	5.2817	9.5364	15.8609	29.1369	38.6100	40.7566
5	5.3012	9.9204	15.9682	29.7633	38.6352	40.7667
6	5.3070	10.1247	16.4221	30.2564	38.6593	40.8174
7	5.6090	10.3198	16.7521	30.4883	39.6679	41.4137
8	5.6866	10.3417	16.8201	31.0909	40.8550	44.8831
9	5.7941	10.4714	16.8314	31.7938	41.0683	46.1311
10	6.3761	10.9570	17.1617	31.8122	41.6499	47.5827

2) 最低功耗比较

当 IP 核数较少时,I^2G 映射算法相对于 IG 映射算法的最低功耗降低幅度较小,在 10%以内. 其中,21 个 IP 核时,I^2G 映射算法比 IG 映射算法的功耗低了 17.46%,这是由于 IP 核数目较少,容易导致 2 种算法均陷入局部最优,因此出现了在 21 个 IP 核时 I^2G 映射算法的功耗降低幅度较大. 从整体趋势来看,随着 IP 核数的增加,功耗降低幅度增加,如图 8 所示. 当 IP 核数为 121 时,I^2G 映射算法比 IG 映射算法功耗降低了 22.64%.

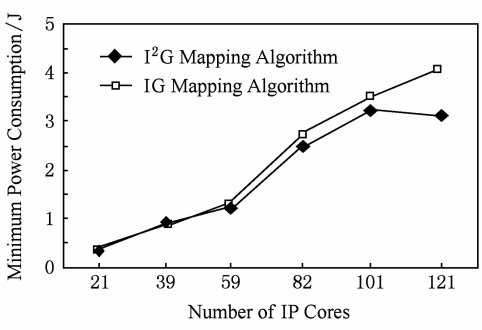

Fig. 8 Comparison of two mapping algorithms' minimum power consumption.

图 8 2 种映射算法最低功耗对比

3) 最高功耗对比

当 IP 核个数为 21 时,I^2G 映射算法相对于 IG 映射算法最高功耗降低了 17.40%. 当 IP 核个数为 82 时,I^2G 映射算法比 IG 映射算法的最高功耗降低了 19.25%(为所做实验的最大降低幅度),如图 9 所示. 当 IP 核数为 121 时,I^2G 映射算法相比 IG 映射算法最高功耗降低了 13.95%.

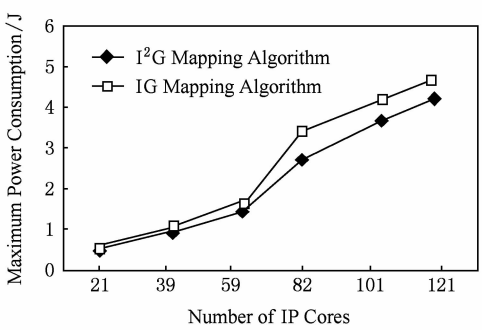

Fig. 9 Comparison of two mapping algorithms' maximum power consumption.

图 9 2 种映射算法最高功耗对比

4) 平均功耗对比

当 IP 核数为 21 时,I^2G 映射算法比 IG 映射算法平均功耗降低了 14.85%;当 IP 核数为 82 时,I^2G 映射算法比 IG 映射算法平均功耗降低了 13.61%;当 IP 核数为 121 时,I^2G 映射算法相比 IG 映射算法平均功耗降低了 17.18%(为所做实验的最大值),如图 10 所示:

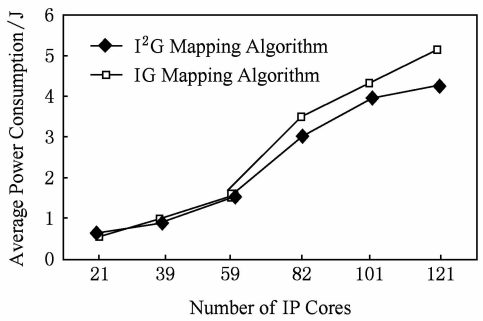

Fig. 10 Comparison of two mapping algorithms' average power consumption.

图 10 2 种映射算法平均功耗对比

4.2.4 种群规模 200 随机任务图运行时间对比

种群规模为 200 时 2 种映射算法的运行时间对比如图 11 所示. 由图 11 可见,I^2G 映射算法比 IG 映射算法的运行时间显著减少;当 IP 核数为 59 时,I^2G 映射算法比 IG 映射算法的运行时间减少了 74.05%.

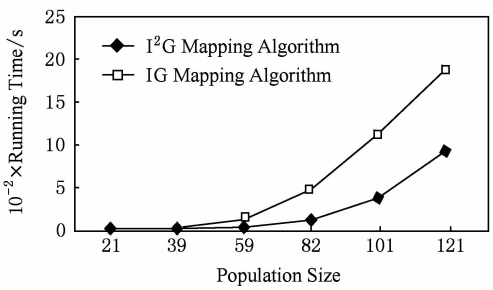

Fig. 11 Comparison of two mapping algorithms' running time with a population size of 200.

图 11 种群规模为 200 的 2 种映射算法运行时间对比

4.2.5 300 种群规模随机任务图功耗与时间对比

1) 2 种映射算法功耗对比

种群规模为 300 时,2 种映射算法平均功耗的比较如图 12 所示. IP 核数为 121 时,I^2G 映射算法比 IG 映射算法的功耗降低了 13.48%.

2) 2 种映射算法运行时间对比

种群规模为 300 时 I^2G 映射算法与 IG 映射算法运行时间对比如图 13 所示. IP 核数为 121 时,I^2G 映射算法比 IG 映射算法的运行时间减少了 85.56%.

4.2.6 随机任务图不同种群规模下运行时间对比

2 种映射算法在 IP 核数为 82 时,种群规模不同时的运行时间对比如图 14 所示. 当种群规模为 600 时,I^2G 映射算法比 IG 映射算法运行时间减少了 89.98%(此时功耗降低 16.36%). 随着种群规模

的增大,I^2G 映射算法的运行时间比 IG 映射算法的运行时间减少幅度逐渐增大;随着种群规模的增加,IG 映射算法的运行时间大幅度增加,而 I^2G 映射算法的运行时间变动幅度很小(这是由于 I^2G 映射算法采用了改进贪心策略生成初始种群,因此,种群规模对映射算法的运行时间影响很小).

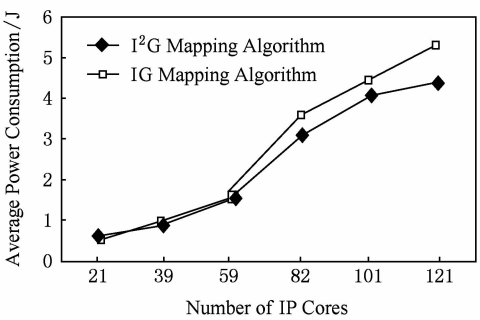

Fig. 12 Comparison of two mapping algorithms' average power consumption with a population size of 300.

图 12 种群规模为 300 时 2 种映射算法平均功耗对比

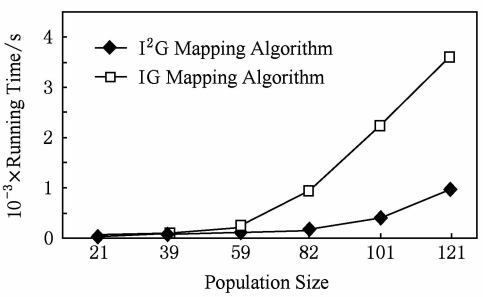

Fig. 13 Comparison of two mapping algorithms' average power consumption with a population size of 300.

图 13 种群规模为 300 时 2 种映射算法平均功耗对比

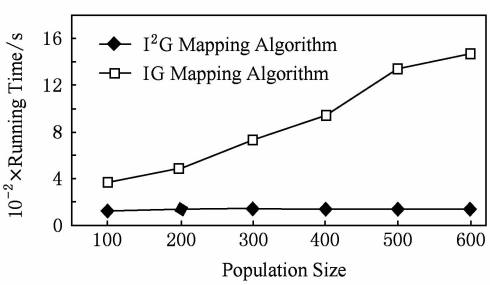

Fig. 14 Comparison of two mapping algorithms's running time with different population sizes and 82 IP cores.

图 14 IP 核数 82 时不同种群规模 2 种映射算法运行时间对比

4.2.7 映射算法收敛速度对比

针对 VOPD 的收敛速度对比如图 15 所示,从迭代开始 IG 映射算法得到的适应值要远远大于基本遗传算法和 I^2G 映射算法,而且在之后的进化过程中,IG 映射算法得到的适应值变化很小,这说明在任务图规模较小时,IG 映射算法容易陷入局部最优.

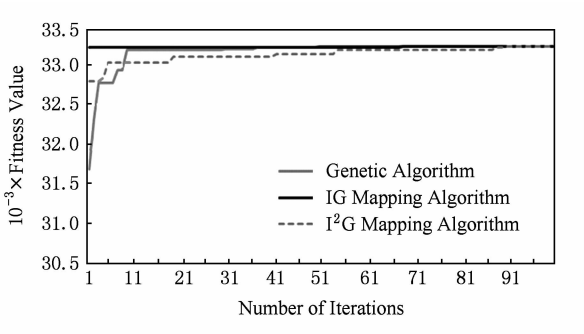

Fig. 15 Convergence speed for two VOPD mapping algorithms.

图 15 针对 VOPD 的 2 种映射算法的收敛速度对比

针对 DVOPD 在 100 次的迭代过程中收敛速度对比如图 16 所示,基本遗传算法的收敛速度比较慢,最终得到的适应值要小于另外 2 种映射算法;IG 映射算法的收敛速度比基本遗传算法快,但是比 I^2G 映射算法慢;I^2G 映射算法在算法运行初期得到的最大适应值要比 IG 映射算法小,而算法运行后期 I^2G 映射算法收敛速度更快,而且得到的适应值更大.

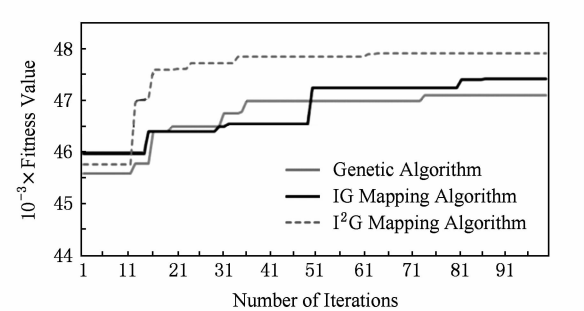

Fig. 16 Convergence speed for two DVOPD mapping algorithms.

图 16 针对 DVOPD 的 2 种映射算法的收敛速度对比

5 结论与展望

5.1 结 论

针对 IG 映射算法运行时间较长的问题,本文提出了基于 I^2G 算法的 3D NoC 低功耗映射算法(简称 I^2G 映射算法).仿真结果表明,随着种群规模的增大,I^2G 映射算法在功耗继续保持降低的前提下,运行时间大幅度地减少.特别是,当 IP 核数为 82、种群规模为 600 时,I^2G 映射算法与 IG 映射算法相比,平均功耗降低了 16.36%,运行时间减少了 89.98%.仿真实验表明,本文所提出的 I^2G 映射算法效果显著.

5.2 展 望

1) 本文采用 XYZ 路由算法进行了 I^2G 映射算法的仿真实验,在今后的研究中将对动态路由策略下的映射问题进行研究.

2) 发热是 3D NoC 设计中需要考虑的重要问题,随着研究的深入,我们将在映射算法研究中加入发热等方面因素的考虑.

3) 随着专用 3D NoC 需求的增多,出现了许多异构 3D NoC 架构,例如基于异构布局、位移交换和基于树图的 3D NoC 架构. 目前,针对异构 3D NoC 架构的映射算法的研究还处于起步阶段,还有很大的发展空间. 在未来的研究中,可以充分利用异构特征,针对不同类型的任务通信图采用不同的映射算法来提高 3D NoC 的映射效率并降低系统的功耗.

4) 目前有一种映射方法是将任务图映射到子网络上,并将子网络用定位的方式在片上网络中找到相应大小的区域,最后将任务图映射到该区域. 采用这样的映射方式,多个任务的映射可同时进行. 但由于任务通信图与子网络上处理单元大小存在不一致性,往往造成子网络上处理单元的浪费。减少处理单元的浪费将是一个值得研究的课题.

参 考 文 献

[1] Lee K, Lee S J, Kim D, et al. Networks-on-chip and networks-in-package for high-performance SoC platforms [C] //Proc of the 2005 Conf on Asian Solid-State Circuits. New York: ACM, 2005: 485-488

[2] Pavlidis V F, Friedman E G. 3-D topologies for networks-on-chip [J]. IEEE Trans on Very Large Scale Integration Systems, 2007, 15(10): 1081-1090

[3] Davis W R, Wilson J, Mick S, et al. Demystifying 3D ICs: The pros and cons of going vertical [J]. IEEE Design & Test of Computers, 2005, 22(6): 498-510

[4] Johnson D S, Garey M. Computers and Intractability: A Guide to the Theory of NP-Completeness [M]. San Francisco, CA: Freeman&Co, 1979

[5] Charles A Q. Thermal-aware mapping and placement for 3-D NoC designs [C] //Proc of the 2005 Int Conf on SOC. New York: ACM, 2005: 25-28

[6] Wang Jiawen. 3D Noc key technology research [D]. Nanjing: Nanjing University, 2012 (in Chinese)
(王佳文. 3D NoC 关键技术研究[D]. 南京: 南京大学, 2012)

[7] Wang Jiawen, Li Li, Wang Zhongfeng, et al. Energy-efficient mapping for 3D NoC using logistic function based adaptive genetic algorithms [J]. Chinese Journal of Electronics, 2014, 23(2): 254-262

[8] Wang Jiawen, Li Li, Hong Bingpan, et al. Latency-aware mapping for 3D NoC using rank-based multi-objective genetic algorithm [C] //Proc of the 2011 Conf on ASIC. New York: ACM, 2011: 413-416

[9] Elmiligi H, Gebali F, El-Kharashi, et al. Power-aware mapping for 3D NoC designs using genetic algorithms [J]. Procedia Computer Science, 2014, 34: 538-543

[10] Ge Fen, Feng Gui, Yu Shuang, et al. Power- and thermal-aware mapping for 3D network-on-chip [J]. Information Technology Journal, 2013, 12(23): 7297-7304

[11] Wang Xiaohang, Liu Peng, Yang Mei, et al. Energy efficient run-time incremental mapping for 3D networks-on-chip [J]. Journal of Computer Science and Technology, 2013, 28(1): 54-71

[12] Zhang Zhen. Research of mapping methods for 3D-mesh-oriented CMP networks on chip [D]. Guangzhou: Guangdong University of Technology, 2012 (in Chinese)
(张振. 基于 3D-MESH 的 CMP 片上网络映射方法研究[D]. 广州: 广东工业大学, 2012)

[13] Qian Yue, Lu Zhonghai, Dou Qiang, et al. Communication performance analysis of the NoCs in 2D and 3D architectures [J]. Computer Engineering & Science, 2011, 33(3): 34-40 (in Chinese)
(钱悦, 鲁中海, 窦强, 等. 片上网络二维和三维结构的通信性能分析[J]. 计算机工程与科学, 2011, 33(3): 34-40)

[14] Matsutani H, Koibuchi M, Amano H. Tightly-coupled multi-layer topologies for 3-D NoCs [C] //Proc of the 2007 Conf on Parallel Processing. New York: ACM, 2007: 75-79

[15] Lin HuaZhou, Zhang DaKun, Huang Cui. Research on low-power mapping for three-dimensional network-on-chip based on improved genetic algorithm [J]. Computer Engineering & Application, 2016, 52(1): 81-88 (in Chinese)
(林华洲, 张大坤, 黄翠. 基于改进遗传算法的 3DNoC 低功耗映射研究[J]. 计算机工程与应用, 2016, 52(1): 81-88)

[16] Pradip K S, Santanu C. A survey on application mapping strategies for network-on-chip design [J]. Journal of Systems Architecture, 2013, 59(1): 60-76

[17] Dick R P, Rhodes D L, Wolf W. TGFF: Task graphs for free [C] //Proc of the 1998 Conf on Hardware/Software Codesign. New York: ACM, 1998: 97-101

Zhang Dakun, born in 1960. Received her PhD degree in computer technology from Northeastern University. Professor at Tianjin Polytechnic University. Her main research interests include 3D networks-on-chip, big data visualization analysis, and virtual reality.

Song Guozhi, born in 1977. Received his PhD degree from Queen Mary, University of London, UK. Associate professor at Tianjin Polytechnic University. His main research interests include 3D networks-on-chip, heterogeneous wireless network integration, and queueing theory (guozhi. song@gmail. com).

Lin Huazhou, born in 1990. MS candidate. His research interests include 3D networks-on-chip (linhuazhou90@126.com).

Ren Shuxia, born in 1973. Received her PhD degree from Tianjin University. Associate professor. Her research interests include 3D networks-on-chip, data mining, and big data analysis (t_rsx@126.com).

2014 年《计算机研究与发展》高被引论文 TOP10

排名	论文信息
1	刘建伟，刘媛，罗雄麟. **玻尔兹曼机研究进展**[J]. 计算机研究与发展，2014，51(1)：1-16 Liu Jianwei, Liu Yuan, and Luo Xionglin. Research and Development on Boltzmann Machine [J]. Journal of Computer Research and Development, 2014, 51(1): 1-16
2	丁兆云，贾焰，周斌. **微博数据挖掘研究综述**[J]. 计算机研究与发展，2014，51(4)：691-706 Ding Zhaoyun, Jia Yan, and Zhou Bin. Survey of Data Mining for Microblogs [J]. Journal of Computer Research and Development, 2014, 51(4): 691-706
3	王静远，李超，熊璋，单志广. **以数据为中心的智慧城市研究综述**[J]. 计算机研究与发展，2014，51(2)：239-259 Wang Jingyuan, Li Chao, Xiong Zhang, and Shan Zhiguang. Survey of Data-Centric Smart City [J]. Journal of Computer Research and Development, 2014, 51(2): 239-259
4	黄冬梅，杜艳玲，贺琪. **混合云存储中海洋大数据迁移算法的研究**[J]. 计算机研究与发展，2014，51(1)：199-205 Huang Dongmei, Du Yanling, and He Qi. Migration Algorithm for Big Marine Data in Hybrid Cloud Storage [J]. Journal of Computer Research and Development, 2014, 51(1): 199-205
5	李晖，孙文海，李凤华，王博洋. **公共云存储服务数据安全及隐私保护技术综述**[J]. 计算机研究与发展，2014，51(7)：1397-1409 Li Hui1, Sun Wenhai, Li Fenghua, and Wang Boyang. Secure and Privacy-Preserving Data Storage Service in Public Cloud [J]. Journal of Computer Research and Development, 2014, 51(7): 1397-1409
6	林闯，董扬威，单志广. **基于 DTN 的空间网络互联服务研究综述**[J]. 计算机研究与发展，2014，51(5)：931-943 Lin Chuang, Dong Yangwei, and Shan Zhiguang. Research on Space Internetworking Service Based on DTN [J]. Journal of Computer Research and Development, 2014, 51(5): 931-943
7	张玉清，王凯，杨欢，方喆君，王志强，曹琛. **Android 安全综述**[J]. 计算机研究与发展，2014，51(7)：1385-1396 Zhang Yuqing, Wang Kai, Yang Huan, Fang Zhejun, Wang Zhiqiang, and Cao Chen. Survey of Android OS Security [J]. Journal of Computer Research and Development, 2014, 51(7): 1385-1396
8	刘雅辉，张铁赢，靳小龙，程学旗. **大数据时代的个人隐私保护**[J]. 计算机研究与发展，2015，52(1)：229-247 Liu Yahui, Zhang Tieying, Jin Xiaolong, and Cheng Xueqi. Personal Privacy Protection in the Era of Big Data [J]. Journal of Computer Research and Development, 2015, 52(1): 229-247
9	蒋卓轩，张岩，李晓明. **基于 MOOC 数据的学习行为分析与预测**[J]. 计算机研究与发展，2015，52(3)：614-628 Jiang Zhuoxuan, Zhang Yan, and Li Xiaoming. Learning Behavior Analysis and Prediction Based on MOOC Data [J]. Journal of Computer Research and Development, 2015, 52(3): 614-628
10	周江，王伟平，孟丹，马灿，古晓艳，蒋杰. **面向大数据分析的分布式文件系统关键技术**[J]. 计算机研究与发展，2014，51(2)：382-394 Zhou Jiang, Wang Weiping, Meng Dan, Ma Can, Gu Xiaoyan, and Jiang Jie. Key Technology in Distributed File System Towards Big Data Analysis [J]. Journal of Computer Research and Development, 2014, 51(2): 382-394

数据来源:中国知网，CSCD;统计日期:2016-02-16