基于遗传算法的NoC处理单元映射研究*)

孙 榕 林正浩
(同济大学微电子中心 上海 200092)

摘 要 传统的基于总线的SoC体系结构及设计方法在解决多处理器的复杂系统中将遇到瓶颈，有效解决方案NoC(片上网络)成为新的发展趋势。本文研究了广泛使用的二维规则型网络(2D-mesh)对影响系统性能的重要因素——功耗建立模型，以及形成处理单元位置映射等问题。最后运用遗传算法来寻找已建立的功耗模型最优解或近最优解。试验结果表明，遗传算法能够使得目标函数很快收敛，起到了很好的全局寻优效果。

关键词 NoC，通讯功耗，处理单元映射，遗传算法

NoC Process Elements Mapping Using Genetic Algorithm

SUN Rong LIN Zheng-Hao
(Microelectronic Center, Tongji Univercity, Shanghai 200092)

Abstract Traditional bus-based SoC architecture and methodology become data exchange bottleneck in complex multi-processor system. New trend of Network on Chip provides a viable solution. 2D-mesh is a widely used NoC topology. This paper studies 2D-mesh network optimization by way of minimizing communication power consumption, which is very import to the system performance. The authors first establish a 2- mesh communication power consumption model, formulate the problem of energy-aware mapping, and then apply genetic algorithm on the model. The result is a fast converged and optimized solution.

Keywords NoC, Power communication power consumption, Process element mapping, Genetic algorithm

1 引言

从现今的微电子发展趋势来看，技术的瓶颈不仅是工艺，还有系统设计。如何集成尽可能多的处理单元于单一芯片之上，使得单一芯片能够尽可能完成更多功能成为设计目标。以PC机体系结构为蓝本的传统SoC体系结构及设计方法在多处理器的超复杂系统将遇到障碍：信号串扰、电源噪音影响了信号的完整性；感应系数作用越来越重要；全局时钟问题越来越严重；多级总线架构的系统满足不了系统扩展性的要求；不同处理单元越来越多意味着各种复杂的接口、协议或者操作系统，使得很难将各处理单元集成到同一芯片上[1]。

芯片的基础构架及其设计方法学的全新变革是必然要发生的，电子信息技术的高速发展要求未来的电子系统能够提供快速解决日益复杂的检测、计算、通信及信号处理的能力，能够提供并行计算和实时处理多任务的能力。NoC (Network on Chip)具备了这些特点，它的提出符合电子信息技术发展的潮流。

2000年11月，瑞典皇家理工学院等单位第一次提出"Network on Chip"的概念。经过几年的发展，NoC发展已经初具规模。各研究机构和研究人员对NoC的理解和提出的架构方案也各有不同，出现了几种片上的网络架构方案：胖树网络(fat-tree)、蜂窝网络(honeycomb)、二维折叠环网络(2D-Folded Tours)、八角网络(octagon)和Kumar等开发的二维规则型网络(2D-mesh)[2]。2D-mesh因其结构和设计简单，容易分析和布局等优点，已成为研究热点。

2 NoC介绍

NoC定义为在一块芯片上，通过交换开关以网络协议为规范实现所有资源的网络互连，互连的资源包括处理器、可编程模块、分布式存储资源及可编程的I/O接口等[3]。资源节点之间通过交换开关将自带寻址信息的数据封包从源地址送到目的地址。NoC提供了资源互连的网络通信构架，各个资源模块硬件设计可以独立于网络构架进行；网络通信平台可升级、可配置，并且适合于不同工作负载的需求。

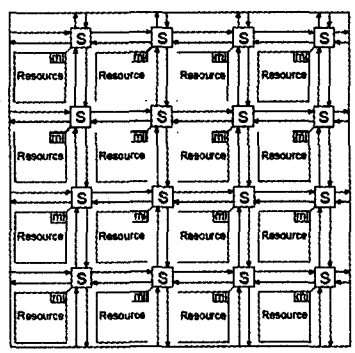


图1 2D-mesh拓扑结构

本文采用2D-mesh拓扑结构，它的布局简单，主要由两部分组成：一是各个相对独立的处理资源(Resource)，处理资源可以是计算单元或者存储单元，或者二者的结合。处理资源通过Rni(Resource network interface)与交换开关直接相

*)国家863高技术研究发展计划资助项目(编号：2005AA1Z1040)。孙 榕 博士生，主要研究方向：SoC设计、NoC基础研究；林正浩 博士生导师，主要研究方向：CPU设计。

连，进而接入网络。二是用于互连的交换开关(Switch)，分布在网络各个节点上，完成数据封包路由和存储转发。如图1，每个交换开关通过4个输入输出双向通道与周围4个交换开关相连，还有一个双向通道与处理资源相连。处理资源与交换开关一一对应，即处理资源与每个节点一一对应。在交换开关的每个输入输出口都设计有缓存器，用于缓存通过交换开关的数据，并设计有硬件路由电路，根据数据封包包头地址信息，来判断数据封包该由哪个出口发出。

NoC是一种时钟全局同步局部异步的构架模式，处理资源进行局部计算，与网络中其他处理资源时钟相对独立；处理资源通过发送接收数据封包实现通信和全局同步。

每一个处理资源都具有唯一地址与交换开关通过Rni相连。任何资源只要配置与芯片上槽兼容的接口，并具有Rni，就能够连入片上网络[3]。参考OSI模型，NoC定义了4个网络协议层：物理层、数据链路层、网络层和传输层。

3 处理单元的位置映射问题的提出

一般的计算机网络是包含多种设备、多种应用的综合平台，而大部分NoC只是针对某一种应用或某几种应用的硬件平台。因此，设计者们对于待设计的具体应用有深刻了解，能够将具体应用分解成由各处理单元完成的具体任务，这些任务能够并行地在NoC上进行。对于已分解的各并行任务，需要找到合适的NoC拓扑结构，也就是确定每个任务应对应网络中的哪个节点。而每个任务都是在特定的处理单元中完成，节点的分配也就成为处理单元的位置映射问题，如图2。处理单元的位置映射，成为NoC设计中的一个重要方面，因为它直接影响到整个片上网络的功耗及其他性能[4]。

这种问题实质上也是一种线性规划问题，是二次分配问题中的一个例子。二次分配问题被认为是NP难问题[5]，搜索空间随系统的规模以阶乘增长，这样的一个问题，对于规模较小的片上网络可以用遍历的方法进行搜索，寻找最佳方案。但是对于规模较大的系统，遍历的方法就显得笨拙费时，可以采取启发式算法来寻找最佳解或者近似的最佳解。在本文中，作者则利用遗传算法这种典型的启发式算法作为搜索工具，来寻找处理单元映射的最佳或近似最佳方案，在后续的文章中将会详细说明。

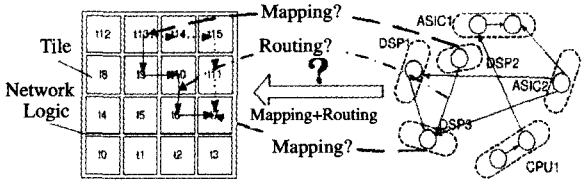


图2 处理单元到NoC拓扑结构的映射

4 通讯功耗模型建立及处理单元位置映射

4.1 与功耗模型建立有关的几个特点

4.1.1 虫洞路由(wormhole routing)算法

虫洞路由[6]能够减少网络缓冲记忆体的数量，使数据封包的延迟不会受到原始端和目的端之间距离所影响，所以被广泛应用于NoC的建置中。采用虫洞路由算法，数据封包被分解为flit(流量控制单元)进行传输，flit的大小与网络参数(如带宽)有关。数据封包的flit-head(头流量控制单元)包含数据的路由信息，用来引导数据封包穿行网络到达目的节点。当flit-head进入网络后，剩余的flit也以流水线方式进入网络，沿着flit-head的路径进行传输，这一方式可以大大减少网络通信的延迟。当flit-head在传输时遇到网络堵塞时，其后的flit只能在当前所在的节点等待，而各节点的交换开关仅需要很小的缓存器就足够可以满足存储需要，而不用设计装下整个数据封包的较大缓存器。

4.1.2 缓存器设计

片上面积的增大意味着成本的提高，因此片上网络的面积资源显得紧张，针对这一问题，片上网络的交换开关与普通数据网络中路由器采用较大的存储设备(如RAM或DRAM)不同，因为片上网络采用了虫洞路由算法，每次暂存的数据由整个数据封包转变为较小的flit，较小的flit的大小一般为几个字节，在文[7]中，flit的大小为4个字节，所以采用寄存器作为暂存数据的缓存器是片上网络的理想之选，这对于应用于某一具体方面的SoC芯片来说，同样也减少了数据存取所花的时延开销。

4.2 通讯功耗模型建立

Ye等人在文[8]中建立了网络路由器的功耗模型，定义单位比特功耗为1比特数据通过路由器传输消耗的能量为

$$E_{bit}=E_{Sbit}+E_{Bbit}+E_{Wbit} \tag{1}$$

E_{Sbit}，E_{Bbit}，E_{Wbit}分别代表开关交换、缓存和连线所消耗的能量。然而，上述模型仅是针对传统意义的数据网络。对于NoC这种结构，需要对该模型进行修改，建立一种新的功耗模型。原因如下：

一般的数据网络遇到拥塞状况时，路由器存储设备(如SRAM或DRAM)需要消耗较大能量来进行数据读取和刷新操作，E_{Bbit}占(1)式的主导地位，而NoC采用寄存器交换开关采用寄存器作为存储设备，不需要消耗太多能量来进行数据缓存，E_{Bbit}不成为(1)式的主导成分。

E_{Wbit}是交换开关内部连线所消耗的能量，对于NoC，交换开关之间的数据通道需要消耗E_{Lbit}，因此从一个交换开关发送1bit数据到相邻交换开关所消耗的平均能量为

$$E_{bit}=E_{Sbit}+E_{Bbit}+E_{Wbit}+E_{Lbit} \tag{2}$$

交换开关之间的连接通道相对比较长(以mm为单位)，E_{Lbit}相对比较大，占(2)式的主导部分。此时，交换开关的寄存器缓存和内部连线所消耗的能量可以近似忽略，式(2)可以近似写成

$$E_{bit}=E_{Sbit}+E_{Lbit} \tag{3}$$

因此，1比特数据从节点r_i传输到相邻节点r_j平均消耗的能量通过(4)式来计算：

$$E_{bit}^{u_i,u_j}=n_{hops}\times E_{Sbit}+(n_{hops}-1)\times E_{Lbit} \tag{4}$$

其中n_{hops}是这一比特数据传输所经过的交换开关数，即经过的节点数。(4)式即是规则2D-mesh结构NoC数据通讯功耗模型。从式中可以看出，数据通讯的功耗与数据经过的交换开关数成正比。假设采用最短路径算法，则数据通讯的功耗就与始点到终点的曼哈顿距离成正比，(4)式可以修改为

$$E_{bit}^{u_i,u_j}=[(d_{r_i,r_j}+1)\times E_{Sbit}+d_{r_i,r_j}\times E_{Lbit}]\times k \tag{5}$$

d_{r_i,r_j}为节点r_i与r_j之间的曼哈顿距离，则有

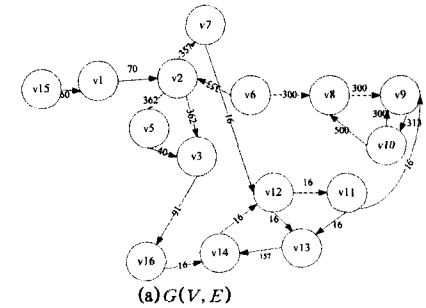
$$n_{hops}=k\times(d_{u_i,u_j}+1)$$

4.3 处理单元位置映射

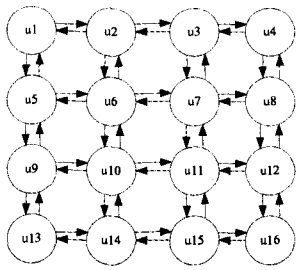
有向图$G(V,E)$构成处理单元任务图，每一节点$v_i\in V$代表每个处理单元，有向边(v_i,v_j)表示为$e_{i,j}\in E$，代表从处理单元v_i到处理单元v_j的数据通讯，$e_{i,j}$的权重则代表从v_i到v_j的通信带宽。有向图$P(U,F)$构成NoC拓扑结构图，节

点 $u_i \in U$ 代表拓扑结构待放处理单元的位置，有向边 (u_i, u_j) 表示为 $f_{i,j} \in F$，代表从节点 u_i 到 u_j 的通信通路，$f_{i,j}$ 的权重就是从节点 u_i 到 u_j 节点通路所能提供的带宽。

图 3(a)为处理单元任务图 $G(V,E)$，(b)为 NoC 拓扑结构图 $P(U,F)$，从 $G(V,E)$ 到 $P(U,F)$ 的映射是一一对应的，即 map:$V \to U$，$map(v_i)=u_j$，$\forall v_i \in V$，$\exists u_j \in U$；且 $|V| \leqslant |U|$。



(a) $G(V,E)$



(b) $P(U,F)$

图 3　处理单元任务图和 NoC 拓扑结构图

如果处理单元 v_i，v_j 分别映射到位置节点 map(v_i)，map(v_j)，假设拓扑结构提供的带宽足够满足需要，则处理单元 v_i 和 v_j 之间的数据通信量即为有向边 $e_{i,j}$ 和 $e_{j,i}$ 的权重之和 $N_{i,j}$(或 $N_{j,i}$)。结合(5)式，处理单元 v_i 和 v_j 之间数据通信消耗的能量为

$$E_{u_i,u_j}=[(d_{u_i,u_j}+1)\times E_{Sbit}+d_{u_i,u_j}\times E_{Lbit}]\times k\times N_{i,j}$$

为使研究方便，上式可以修改为

$$E_{u_i,u_j}=[d_{u_i,u_j}\times(E_{Lbit}+E_{Sbit})]\times N_{i,j}=d_{u_i,u_j}\times E_{bit}\times N_{i,j}$$
$$=d(map(v_i),map(v_j))\times e(map(v_i),map(v_j)) \quad (6)$$

其中 $E_{Lbit}+E_{Sbit}=E_{bit}$ 为单位比特数据从经过相邻两个交换开关所消耗的总能量。$d(map(v_i),map(v_j))$ 为 $map(v_i)$ 与 $map(v_j)$ 的曼哈顿距离，$e(map(v_i),map(v_j))$ 是处理单元 v_i 与 v_j 之间通讯的数据传输单位距离所消耗的能量，如果 v_i 与 v_j 之间没有数据通讯，则 $e(map(v_i),map(v_j))$ 为零。

所以，关键是要找到一种映射方式，使 $G(V,E)$ 映射到 $P(U,F)$，满足

$$\min\{\sum_{\forall v_i,v_j\in V} d(map(v_i),map(v_j))\times e(map(v_i),map(v_j))\}$$

这也是本文寻找的目标。

5　基于遗传算法寻找最优解

5.1　基本思想

类似于其他寻找最优组合的问题，寻找从 $G(V,E)$ 到 $P(U,F)$ 的映射最优解，是一类 N-P 难问题。本文的目的是要寻找一个或者一些能够在性能和寻优计算时间上获得平衡的方案。遗传算法在这两方面表现出它的优势。

5.2　算法实现

随机产生有效的解决方案作为初始种群(初始种群含 n 个个体成员)，初始种群的每个成员就是一条染色体，代表着一种映射解决方案。每条染色体由一串基因产生，每个基因代表 NoC 结构的一个节点，它的值则是处理单元集合中被选择映射到该节点的处理单元编号。

计算初始种群每个成员的适应性函数：

$$f=\sum_{\forall v_i,v_j\in V} d(map(v_i),map(v_j))\times e(map(v_i),map(v_j))$$

从上一代中产生比较好的子代，采用单点交叉法中的非常规交叉方法，随机选择一个交叉位，两个后代交叉位之前的基因分别继承双亲的交叉位之前基因，交叉位之后的基因分别按照异方基因顺序选取不重基因。变异采用单点非常规法，变异点上的基因与染色体上其他基因重复，非变异点上的这个重复基因需要替换成变异点上未改变前的基因。

循环地按照上述方法产生新的子代，直到循环次数达到设定要求。

从产生的最后一代中找出较好的子代，作为进一步研究的对象。

6　实验

作者利用 matlab 工具实现了上述寻优过程，在本文中选用了非常规交叉法和变异法，修改了 matlab 自带的遗传算法，其中非常规交叉法的 matlab 语言描述如下：

```
function
[child]=cross(parent1,parent2,xOverPoint)
parent1_L = length(parent1);
parent2_L = length(parent2);
parent1_tmp =
parent1([xOverPoint+1:parent1_L]);
parent2_tmp =
parent2([xOverPoint+1:parent2_L]);
[child_tmp,parent1i,parent2i]=
setxor(parent1_tmp,parent2_tmp);
parent2_tmp([parent2i]) =
parent1_tmp([parent1i]);
child =[parent1([1:xOverPoint]),parent2_tmp];
```

作者随机产生集合作为 $\{d|d(map(v_i),map(v_j)),\forall v_i,v_j\in V\}$，并随机产生对应的 $\{e|e(map(v_i),map(v_j)),\forall v_i,v_j\in V\}$，上述两个集合作为算法的输入，输出为处理单元编号的一种排列。通过 matlab 工具进行算法仿真，当算法循环计算到达已设定的子代数，算法停止。

结论　利用 matlab 工具对每一代中最优个体的适应性值和每代个体的平均值进行捕捉，得到图 4。从图中可以看到适应性函数能够很快收敛，遗传算法能够在较短的时间进行全局寻优，可以选择最后一代中适应性函数较好的个体作为最优或近最优解。

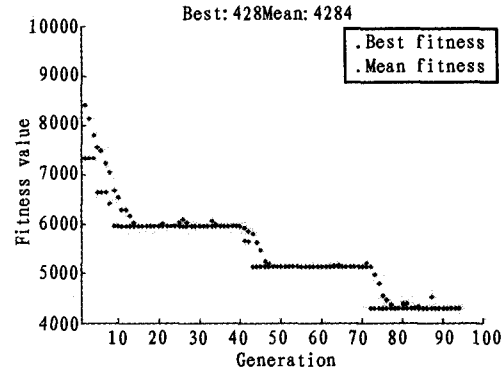


图 4　适应性函数收敛图

(下转第 84 页)

包或乱序概率 p。当 $R\ \%\ 100 < p * 100$ 时，在概率范围内，即此时主动丢包或主动乱序。模拟丢包可以简单地放弃发送实现。对于模拟乱序情况，在发送的时候将需传输乱序的数据包缓存起来，然后将它延后该数据包原发送位置 W 的位置后进行发送（W 为乱序偏差），来模拟数据包传输乱序。

本次测试在 CPU 主频 1.70GHz、内存 256MB 环境下进行。测试时间为 1min，发送帧速率 15fps，最大关键帧间隔为 15，缓冲区缓存长度为 6 帧。测试丢包情况设置图像分辨率为 352×288，图 4 为其测试结果；测试乱序情况分别取乱序偏差值为 5、10、15，并分别测试图像分辨率为 176×144 和 352×288 两种情况，其测试结果如图 5、图 6 所示。

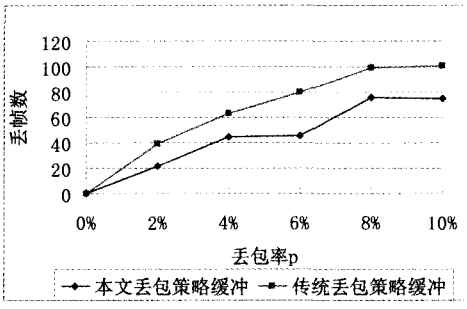


图 4　丢包测试结果

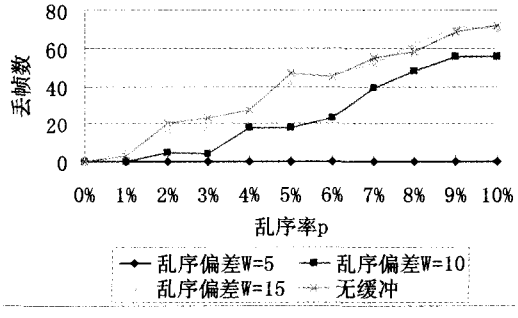


图 5　乱序测试结果（176×144）

从图 4 中可以看出，新缓冲增加了判断非参考帧策略，避免了一些不必要的丢帧，减少了一定数量的丢帧数。由图 5 可知，在乱序偏差较小的情况下，本缓冲设计有较好的性能，但是随着乱序偏差增大，性能提高不明显。这是由于在乱序偏差较大的情况下，乱序包所在帧的间隔可能会大于缓冲区缓冲长度 6 帧。这样，缓冲区将认为乱序包不在缓冲范围之内，丢弃乱序包。而在图 6 所示的较大分辨率下，由于每一帧的数据量较大，分包较多，缓冲区暂存了 6 帧，即暂存了更多的数据包，因此可以适应更大的乱序偏差。综合可知缓冲区性能与图像分辨率和缓冲区长度有关。但是整体来看仍然有不错的性能指标，并且缓冲区由时间驱动，精确控制帧数据向解码器传递，同步了发送端和接收端的视频数据，平滑了图像显示，具有良好的主观评测性能。此外，该缓冲区减少了一次拷贝，节省了相当可观的系统资源。

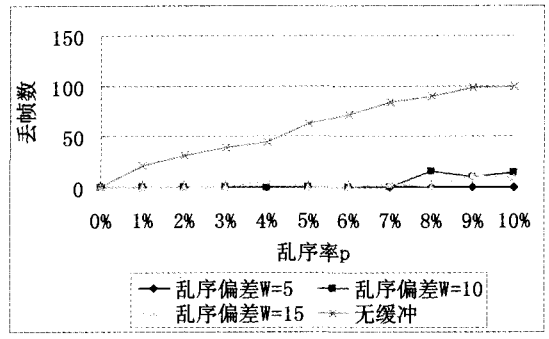


图 6　乱序测试结果（352×288）

结论　实时视频流应用已经越来越广泛，对接收端缓冲的性能要求也越来越高。本文提出了一种基于时间驱动的具有同步功能的缓冲区设计方案。仿真测试表明，该方案有良好的客观性能指标和较好的主观感觉，已应用于秦皇岛东大软件视频会议系统、北京网通网上直播系统等实时视频流接收客户端的设计中，取得较好的效果。

参 考 文 献

1　Richardson I E G. Video Codec Design[M]. Wiley, 2002. 27～46

2　Casner S, Frederick R, Jacobson V, et al. RFC3550, RTP[S]. USA.: Columbia University, 2003. 13～52

3　Liu Jing, Niu Zhisheng. An Adaptive Receiver Buffer Adjust Algorithm for Voice & Video on IP Applications[C]. In: 2005 Asia-Pacific Conference on Communications, Perth, Western Australia, October 2005. 669～673

4　Zhou Yuanyuan, Chen Zhifeng, Li Kai. Second-level Buffer Cache Management[J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 15: 505～519

5　毕厚杰. 新一代视频压缩编码标准——H. 264/AVC[M]. 人民邮电出版社, 2006

6　Richardson I E G. H. 264[S]. Switzerland: ITU-T, 2005. 93～184

7　Wenger S H. 264/AVC Over IP[J]. IEEE Trans Circuits Syst. Video Technol, 2003, 13: 645～656

（上接第 53 页）

参 考 文 献

1　Kumar S, Jantsch A, Soininen J, et al. A Network on Chip Architecture and Design Methodology[C]. In: Proceedings of IEEE Computer Society Annual Symposium on VLSI, Washington DC, USA: IEEE Computer Society, 2002

2　Wang H S, Zhu X, Peh L S, et al. Orion: a power-performance simulator for interconnection networks[C]. In: Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture, Washington DC, USA: IEEE Computer Society, 2002. 294～305

3　Hemani A. Jantsch A, Kumar S, et al. Network on a chip: An architecture for billion transistor era[C]. In: Proceeding of the IEEE NorChip Conference, November 2000

4　Hu J, Marculescu R. Energy-aware Mapping for Tile-based NoC Architectures Under Performance Constraints[C]. In: Proceedings of the 2003 onference on Asia South Pacific Design Automation. New York, USA: ACM Press, 2003. 233～239

5　Garey M R, Johnson D S. Computers and intractability: a guide to the theory of NP-completeness[M]. New York, USA: W H Freeman & Co, 1979

6　Hu Jingcao, Marculescu R. A survey of wormhole routing techniques in direct networks[C]. Los Alamitos, CA, USA: IEEE Computer Society Press, 1993

7　Khan G N, Gu Wei. Fault-tolerant Wormhole Routing Using a Variation of Distributed Recovery Block Approach[C]. IEEE Proceedings of Computers and Digital Techniques, 2000, 147(6): 397～402

8　Ye T T, Benini L, De Micheli G. Analysis of power consumption on switch fabrics in network routers[C]. In: Proceedings of Design Automation Conference, June 2002. 524～529

9　孙增圻，张再兴，邓志东. 智能控制理论与技术[M]. 北京：清华大学出版社，1997

10　张传顺，莫蓉，石胜友，等. 基于遗传算法的制造网格资源调度方法研究[J]. 中国机械工程，2006, 17(18)