树拓扑片上网络的低能耗映射

常政威[1]，桑 楠[2]，熊光泽[2]

(1. 四川电力试验研究院 成都 610072; 2. 电子科技大学计算机科学与工程学院 成都 610054)

【摘要】针对树拓扑片上网络(NoC)中通信时延受约束的低能耗映射问题，提出了一种递归的二路划分算法RPM(recursive bipartitioning for mapping)。RPM基于分而治之策略，首先将NoC映射转化为多层次的IP核通信任务图划分问题，并采用带参数的Kernighan-Lin算法实现最小割值划分。实验结果表明，与已有算法相比，RPM可以在较短的时间内获得能耗更低的映射解。通过设置不同的参数，RPM既可以用于生成高质量的优化解，也可用于快速的NoC设计空间探索中。

关 键 词 二路划分; 能耗优化; 映射; 片上网络; 树拓扑

中图分类号 TP302 **文献标识码** A doi:10.3969/j.issn.1001-0548.2010.04.029

Low Energy Mapping for Tree Based Networks-on-Chip

CHANG Zheng-wei[1], SANG Nan[2], and XIONG Guang-ze[2]

(1. Sichuan Electric Power Tests and Research Institute Chengdu 610072 ;

2. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract A recursive bipartitioning algorithm, RPM, is proposed for low energy mapping in tree-based Network-on-Chip (NoC) architectures subject to communication latency constraints. The mapping problem is formulated to multi-level IP core communication task graph partitioning problems, and the modified Kernighan-Lin mincut heuristic is used to solve them. Experimental results show that RPM obtains lower energy mapping solutions compared with existing algorithms.

Key words bipartitioning; energy optimization; mapping; networks-on-chip; tree based topology

随着集成电路技术的快速发展，未来单个芯片上的晶体管数目将会达到数十亿个，包含几十个或上百个知识产权(intellectual property，IP)核的片上系统(system-on-chip，SoC)成为高性能嵌入式系统的发展趋势。为实现众多微处理器、DSP、存储器和I/O模块等IP核之间的高效数据通信，SoC迫切需要可扩展、高带宽和并行的通信架构。片上网络[1](network-on-chip，NoC)是为解决深亚微米时代复杂的片上多核通信问题而出现的，是当前SoC及相关领域的一个研究热点。对于NoC而言，能耗已成为首要的优化因素之一。

映射是NoC设计中非常重要的一个步骤。在确定了SoC所选用的IP核之后，NoC映射将系统中用于计算和存储功能的所有IP核分配到给定网络拓扑中的不同路由器端口，在通信性能约束条件下，使目标函数(如通信能耗)最优。一般来说，对于包括n个IP核的SoC，共有个$n!$个不同的映射[2]。文献[3]指出NoC映射等同于受约束的二次分配问题，用分支限界法求解带宽约束下的能耗最优NoC映射问题，与随机映射相比节省了约60%的通信能耗。为降低算法的求解时间，文献[3]还采用监控优先队列长度的启发式方法，不过是以提高NoC能耗为代价的。其他的低能耗NoC映射算法见文献[4-6]。

由于二维Mesh具有简单直观、易于芯片平面布局等特点，在已有的NoC映射算法中[3-6]，大都采用此类网络拓扑。但是，网格类的拓扑结构主要适用于通用的、包括许多同构处理核的多核或众核CPU架构。嵌入式系统具有专用性，组成SoC的各IP核通常是异构的，在通信需求上也是不对称的，需要采用其他更适合的网络拓扑。

与已有的研究工作不同，本文主要解决基于树拓扑的低能耗NoC映射问题。与二维Mesh相比，树更适合于呈现出通信不对称和流量局部性的SoC[7-8]，如SPIN[9]。并且，与其他网络拓扑相比，树在硬件资源上是最有效的[7-8]。另一方面，树已用于实现层次化NoC中的子网[10]，子网内的NoC映射处于NoC

收稿日期：2008 – 11 – 17；修回日期：2009 – 04 – 13
基金项目：国家863项目(2006AA01Z173, 2007AA01Z131)
作者简介：常政威(1981 –)，男，博士，主要从事嵌入式实时系统、设计自动化及低能耗设计方面的研究.

体系结构设计步骤的内层，在设计和优化过程中需要多次调用，更需要高效的优化算法。

本文面向对通信时延有严格要求的实时SoC应用和基于树的NoC拓扑，定义了能量感知的NoC映射问题，并提出了一种采用递归二路划分的求解算法。实验结果表明，本文算法优于已有的分支限界算法，并具有较很快的运行速度。

1 NoC映射问题

本节首先定义基于完全二叉树的NoC拓扑结构及其相关的能耗模型，然后给出NoC映射问题的形式化定义。需要说明的是，虽然本文中的NoC映射问题采用二叉树拓扑，但对于多叉树、胖树[9]等稍做调整也能适用。

1.1 NoC拓扑简介

如图1所示，设NoC的拓扑结构为一棵完全二叉树，包括$n=2^h$个叶子节点和$n-1$个内节点。叶子是用于计算和存储的处理节点，每个处理节点可放置一个IP核。其余的内节点是用于通信的路由器，路由器之间通过链路连接。

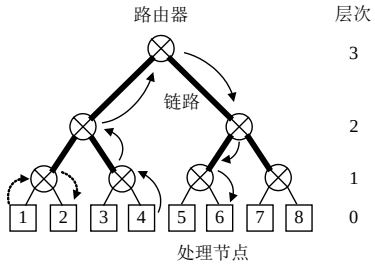


图1 二叉树NoC拓扑

树内的节点是分层次组织的。具体来说，叶子节点位于第0层，根节点位于最高层即第h层。例如，图1中包括8个处理节点，$h=3$。除根节点外，每个位于第i层的路由器分别连接至第$i+1$层的父节点，以及两个$i-1$层的孩子节点。

网络内的路由非常简单，来自处理节点的通信数据首先向相连接的路由器即它的父节点转发，当路由器节点收到一个数据包时，若该数据包的目的节点位于它的子树中，则向下层中相应的孩子节点转发，否则继续向其父节点转发，数据包会从叶子节点开始，不断向上一层的路由器转发，然后在某个路由器节点改变传输方向，逐步向下传输至另一个叶子节点。

因此，任意两个处理节点之间的最短路径所经过的路由器是完全相同的。本文对每条路由路径也定义一个层次level，表示路径中唯一的一个层次最高的路由器的层次。

例如，在图1中，编号为1和2的处理节点之间通信，只需要通过位于第1层的一个路由器，该路径的层次为1。而对于编号为4和6的处理节点之间的通信，则必须通过根节点，路径的层次为3。

显然，层次为level的路径中包括(2level−1)个路由器，即路由距离为(2level−1) hops。

1.2 能耗模型

当数据包在NoC中传递时，路由器和链路都会产生动态能耗。首先考虑两个兄弟叶子节点之间的通信，例如图1中的节点1和2，路由路径的层次为1，沿该类路径传递1 bit数据平均消耗的能量为：

$$E_1 = E_{R_{\text{bit}}} \tag{1}$$

式中 $E_{R_{\text{bit}}}$ 表示单个路由器传输1 bit数据的平均动态能耗[3]。

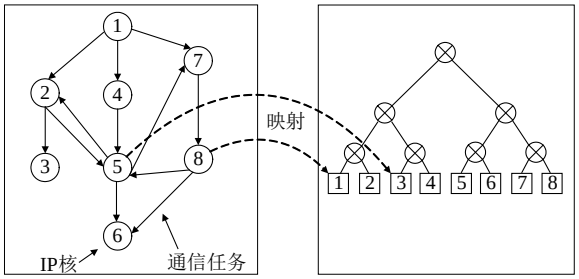
一般来说，通过层次为level的路由路径传递1 bit数据平均消耗的能量E_{level}为：

$$E_{\text{level}} = E_{R_{\text{bit}}} + 2E_{L_{\text{bit}}} + 2E_{\text{level}-1} = (2\text{level}-1)E_{R_{\text{bit}}} + 2(\text{level}-1)E_{L_{\text{bit}}} \tag{2}$$

式中 $E_{L_{\text{bit}}}$ 表示单条链路传输1 bit数据的平均动态能耗[3]。因此，两个节点间的通信流量保持不变时，对应的路由路径层次越高，耗费的能量也越多。

1.3 NoC映射问题定义

如图2所示，NoC映射就是将IP核通信任务图中的每一个IP核分配至NoC拓扑中唯一的一个处理节点，使其与某一个路由器端口直接连接，在满足通信任务图中每个通信任务时延约束的条件下，使得NoC通信能耗最小。



a. IP核通信任务图 b. NoC拓扑结构

图2 NoC映射问题

本文对NoC映射作如下形式化的定义。

定义 1 IP核通信任务有向图CTG(C, A)中，每个顶点$c_i \in C$表示一个IP核；每条有向边$a_{ij} \in A$表示从c_i到c_j的通信任务；权重$v(a_{ij})$表示通信任务a_{ij}的数据流量，单位为bit；权重$t(a_{ij})$表示通信任务a_{ij}允许的最大传输时延，用hops表示。

定义 2 NoC体系结构有向完全图NAG(R,P)中，每个顶点$r_i \in R$表示一个处理节点；每条有向边$p_{ij} \in P$表示从节点r_i到节点r_j的路由路径；$l(p_{ij})$是路径p_{ij}的层次；$d(p_{ij})$表示节点r_i与节点r_j间的路由距离，用hops表示，且$d(p_{ij})=2l(p_{ij})-1$。

给定IP核通信任务图CTG和NoC体系结构图NAG，且有$|C|=|R|$，通信时延受约束的低能耗NoC映射问题(简称为NoC映射)就是要寻求一个一一映射：

$$\phi: C \to R \Rightarrow r_i = \phi(c_i), \forall c_i \in C, \exists r_i \in R,$$
$$\forall c_i \neq c_j \in C, \phi(c_i) \neq \phi(c_j)$$

使NoC通信能耗最优为：

$$\min\left\{E(\phi)=\sum_{\forall a_{ij}\in A}(E_{l(p_{\phi(c_i),\phi(c_j)})}\times v(a_{ij}))\right\} \quad (3)$$

$$\text{s.t.} \quad \forall a_{ij} \in A, t(a_{ij}) \geqslant d(p_{\phi(c_i),\phi(c_j)}) \quad (4)$$

式(4)表示任意通信任务的时延必须得到满足。

2 NoC映射算法

在已有的映射算法中，大都将NoC映射看作二次分配问题[3-6]，用相应的启发式或智能优化算法求解，效率较低。针对树拓扑NoC映射的特点，本文将其转化为IP核通信任务图的划分问题，基于分而治之的求解策略，采用一种递归的二路划分算法，不断地将通信任务图的子图分配到NoC拓扑的子树中，直到任务图中只包括两个IP核且对应的子树中只包含一个路由器，最终将每个IP核映射到唯一的处理节点。

2.1 将NoC映射转化为划分问题

由式(3)可将NoC映射的目标函数转化为：

$$E(\phi)=\sum_{\text{level}=1}^{h}\left(E_{\text{level}}\times\left(\sum_{\forall a_{ij}\in A\wedge l(p_{\phi(c_i),\phi(c_j)})=\text{level}} v(a_{ij})\right)\right) \quad (5)$$

因此，优化NoC通信能耗$E(\phi)$只需要减少不同层次路由路径上的累积通信流量，尤其是减少高层次路径上的通信流量。

2.1.1 最小割划分

下面介绍IP核通信任务图CTG和NoC体系结构图NAG如何在不同层次上进行划分，并最小化划分间的割值(通信流量)，达到优化通信能耗的目的。

令NAG_1= NAG，$n=|R|$为NAG中的处理节点数，则$h=\log_2|R|$为P中所有路由路径的最高层次。按如下规则得到NAG的$n-1$个子图(NAG_1, NAG_2,…, NAG_{n-1})。

规则 1　NAG_{2k}和NAG_{2k+1}是NAG_k ($1 \leqslant k \leqslant n/2-1$)的一个平衡二路划分，即$R_k = R_{2k} \cup R_{2k+1}$，$R_{2k} \cap R_{2k+1} = \Phi$。

规则 2　P_{2k}和P_{2k+1}中路由路径的最高层次为$h-\lfloor\log_2 k\rfloor-1$，$R_{2k}$和$R_{2k+1}$间的路由路径的层次均为$h-\lfloor\log_2 k\rfloor$。

因此，NAG_{2k}和NAG_{2k+1}恰好是NAG_k所对应NoC的两棵子树的拓扑图。NAG的$n-1$个子图对应于$n-1$个子网，NAG_k是一个层次为$h-\lfloor\log_2 k\rfloor$的完全二叉树，R_{2k}和R_{2k+1}间的路由距离为$2(h-\lfloor\log_2 k\rfloor)-1$ hops。

2.1.2 NoC映射的转化

令CTG_1=CTG，则NoC映射可以转化为下面的$n/2-1$个划分问题。

将CTG_k划分为两个相等的子图CTG_{2k}和CTG_{2k+1}，即$C_k = C_{2k} \cup C_{2k+1}$，$C_{2k} \cap C_{2k+1} = \Phi$；且将$CTG_k$映射至$NAG_k$，即$\forall c_i \in C_k$，$\phi(c_i) \in R_k$；并使得两个划分间的通信流量之和即割值$\text{Cut}_k$最小为：

$$\min\left\{\text{Cut}_k=\sum_{\forall c_i\in C_{2k},\forall c_j\in C_{2k+1}} v(a_{ij})\right\} \quad k=1,2,\cdots,\frac{1}{2}n-1 \quad (6)$$

$$\text{s.t.}\ \forall c_i \in C_{2k} \wedge c_j \in C_{2k+1}, t(a_{ij}) \geqslant 2(h-\lfloor\log_2 k\rfloor)-1 \quad (7)$$

最终，对于包括两个IP核的CTG_k ($n/2 \leqslant k \leqslant n-1$)，只需要将它们一一映射至$NAG_k$的两个处理节点，$\text{Cut}_k$即为两个IP核间的通信流量。

映射完成后，NoC通信能耗为：

$$E(\phi)=\sum_{\text{level}=1}^{h}\left(E_{\text{level}}\times\sum_{k=2^{h-\text{level}}}^{2^{h-\text{level}+1}-1}\text{Cut}_k\right) \quad (8)$$

2.2 递归二路划分算法

图的划分本身也是NP hard问题，由于在NoC映射过程中要不断地进行任务图的划分，必须采用高效的二路划分算法，并且在划分过程中需考虑通信任务的时延约束。

2.2.1 最小割划分的实现

文献[11]提出了一个用于图的最小割二路划分算法，以下简称为KL。KL最初来源于电路划分问题，用于将电路网表划分为相等的两个部分，以最小化它们之间的连线数量，即割值。

本文使用的最小割划分基于KL实现。在划分CTG_k时，首先将C_k中的IP核随机分为两个集合，如前面所述，将割值定义为两个集合间的通信任务的流量之和。

根据问题的特点，还对KL作如下改进。

(1) 对CTG_k进行划分时，由于NAG_{2k}和NAG_{2k+1}之间的通信时延为$2(h-\lfloor \log_2 k \rfloor)-1$，对于通信任务$a_{ij} \in A_k$，若$t(a_{ij}) < 2(h-\lfloor \log_2 k \rfloor)-1$，则$c_i$与$c_j$必须位于同一个划分中，为此将$c_i$与$c_j$聚簇，看作一个超IP，一个超IP中可能会包括多于2个的IP核。在KL的实现中，一个超节点作为一次移动的基本单位，其内部包括的IP核不能被单独移动。

(2) 由于超节点的存在，节点在两个划分之间的移动采用单向移动，在KL一次迭代的中间步骤，两个划分可能不再保持平衡。解决办法是，若最大的超IP中包括m个IP核，则两个划分的大小最多只能相差m，但一次迭代结束后仍然选择平衡的划分状态。

2.2.2 映射算法的实现

由于KL是从一个随机的划分开始，由以上最小割迭代过程获得的解可能会较差。另外，划分所处的层次越高，割值对NoC通信能耗的影响也越大，而且子图的规模也越大。基于以上观察，可以根据不同的层次，改变调用基于KL的最小割划分过程的次数。

定义如下重启函数：

$$\text{runs}(\text{level}, \text{rt}) = \text{level}^{\text{rt}}, \quad \text{rt} = 0,1,2,\cdots \tag{9}$$

式中 level= $h-\lfloor \log_2 k \rfloor$是$CTG_k$映射到的子网$NAG_k$的层次，rt是用于实现运行时间与求解质量折中的参数。runs(level, rt)用于控制对CTG_k调用最小割划分的次数，当rt不变时，level越大，则runs(level, rt)的值也越大。

算法1给出了求解NoC映射的递归二路划分算法，称为RPM(recursive bipartitioning for mapping)。输入CTG、NAG和rt，RPM得到映射函数ϕ及n–1个割值Cut_k，对应的NoC通信能耗可以由式(8)计算。

算法 1 RPM

输入：CTG_k、NAG_k、rt；

输出：ϕ、Cut_k。

level = $\log_2 |R_1| - \lfloor \log_2 k \rfloor$； //$NAG_k$的层次为level

```
if (level==1){
  /*将 C_k 中两个 IP 核映射到 R_k 的节点中*/
  φ(C_k) = R_k ;
  Cut_k = v(A_k) ; // v(A_k) 为 A_k 内的通信流量
  return;
        //根据 NAG_k 得到 NAG_2k 和 NAG_2k+1
  (NAG_2k,NAG_2k+1)=subtree(NAG_k) ; // 根据 NAG_k 得到 NAG_2k 和 NAG_2k+1
  CCTG_k = cluster(CTG_k); //根据时延约束聚簇
  Cut_k = mincut(CCTG_k); //调用最小割过程 KL
  /*返回割值最优的划分 CTG_2k 和 CTG_2k+1*/
  for round=2 to runs(level,rt) {
  if (Cut_k<mincut(CCTG_k))
      Cut_k=mincut(CCTG_k)
      /*递归调用 RPM*/
  RPM(CTG_2k,NAG_2k,rt);
  RPM(CTG_2k+1,NAG_2k+1,rt);
return;
```

当level =1时，RPM算法中第2～7行时间复杂度为$O(1)$。当level >1时，第8行subtree(NAG_k)只需要遍历P_k中的所有路由路径，生成两个NoC体系结构子图NAG_{2k}和NAG_{2k+1}，时间复杂度为$O(|R_k|^2)$；第9行聚簇过程cluster(CTG_k)需要遍历A_k中的所有通信任务，将相应通信时延约束较强的多个IP核标记为一个超IP，时间复杂度为$O(|C_k|^2)$；基于KL的最小割划分过程mincut($CCTG_k$)的时间复杂度为$O(|C_k|^3)$[11]，在10～14行共执行$(\log_2 n - \lfloor \log_2 k \rfloor)^{\text{rt}}$次，其中$R_k$=$C_k$= $2^{h-\lfloor \log_2 k \rfloor}$，$n = |R|$。最终，可得到递归的RPM算法的时间复杂度为$O(n^3 \log^{\text{rt}} n)$ [12]。

3 实验结果

为验证本文算法的有效性，将其应用于不同类型和规模的NoC映射问题中。算法用C++编写，基于Intel P IV 2.6 GHz处理器，在Windows 2 000操作系统下运行。RPM的参数rt分别取0、1和2，3种不同的算法实现简记为RPM_{rt}，即RPM_0、RPM_1和RPM_2。

实验中使用的IP核通信任务图来源有：(1) 视频/音频应用[3]部分，H263编码器(8核)，H263编码器&MP3编/解码器(16核)，对应的通信任务图见文献[6]；(2) 包括32和64个IP核的随机应用部分，分别将以上通信任务图映射到包括相同处理节点的NoC平台中。

用RPM_{rt}和启发式分支限界(branch-and-bound，BB)[3]算法分别求解4个不同规模的NoC映射问题。对于同一个问题，rt较大的RPM_{rt}内部调用KL的次数较多，需要更多的CPU运算时间。为公平起见，对每个求解问题，BB的运行时间不少于RPM_2。

与BB相比，RPM_{rt}的能耗优化结果如表1所示。对于包括8个IP核的NoC应用，所有算法的结果基本相同。但随着问题规模的增加，rt值越大的RPM_{rt}优势越明显。当IP核的个数大于16时，即使RPM_0也优

于BB。对于64个IP核的NoC应用，RPM_0、RPM_1和RPM_2与BB相比，分别节能7%、11%和16%。

表1　RPM_{rt}的能耗优化结果比较

IP核数目	$\frac{RPM_0}{BB}$ /(%)	$\frac{RPM_1}{BB}$ /(%)	$\frac{RPM_2}{BB}$ /(%)
8	100.0	100.0	100.0
16	99.9	99.9	97.8
32	96.1	91.8	91.8
64	92.8	89.1	84.0

不同参数的RPM算法求解各问题的运行时间如图3所示。RPM的速度非常快，对于规模为64的映射问题，RPM_2只需要不到0.5 s的求解时间。

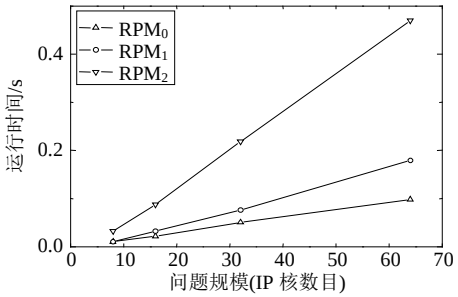


图3　RPM_{rt}的运行时间

另外，从表1和图3中可以看出RPM算法求解质量与运行时间之间的关系。参数rt较大的RPM的优化结果最好，但比较耗时，对于大规模的NoC映射问题，可以采用RPM_2以获得高质量的解。虽然RPM_0的结果相对较差，但其运行最快，而且对于小规模问题，其性能与RPM_1、RPM_2相当。因此，RPM_0可以嵌套在层次化NoC的子网映射或其他的IP核选择、任务指派等设计步骤内部，用于快速地设计空间探索。

4　结　论

针对通信时延约束的低能耗树拓扑NoC映射问题，提出一种高效的求解算法RPM。实验结果表明，RPM可以在较短的运行时间内获得性能优异的NoC映射解，既可用于获得高质量的优化结果，也可用于NoC设计空间探索的快速内部迭代。

本文的讨论主要限于二叉树NoC拓扑，但对于m叉树(m>2)，文中的算法仍然适用。在该类情况下，RPM算法中的最小割过程采用多路划分，仍然可以基于KL实现通信任务图的m路划分[11]。

参　考　文　献

[1] ATIENZA D, ANGIOLINI F, MURALI S, et al. Network-on-chip design and synthesis outlook[J]. Integration, the VLSI Journal, 2008, 41(3): 340-359.

[2] MARCON C A M, MORENO E I, CALAZANS N L V, et al. Comparison of network-on-chip mapping algorithms targeting low energy consumption[J]. IET Computers & Digital Techniques, 2008, 2(6): 471-482.

[3] HU J, MARCULESCU R. Energy-aware mapping for tile-based NoC architectures under performance constraints [C]//Proceedings of the Conference on Asia South Pacific Design Automation. New York, USA: ACM Press, 2003: 233- 239.

[4] MARCON C A M, MORENO E I, CALAZANS N L V, et al. Comparison of network-on-chip mapping algorithms targeting low energy consumption[J]. IET Computers & Digital Techniques, 2008, 2(6): 471-482.

[5] 杨盛光, 李　丽, 高明伦, 等. 面向能耗和延时的NoC映射方法[J]. 电子学报, 2008, 36(5): 937-942.
YANG Sheng-guang, LI Li, GAO Ming-lun, et al. An energy- and delay-aware mapping method of NoC[J]. acta electronica sinica, 2008, 36(5): 937-942.

[6] 常政威, 谢晓娜, 桑　楠, 等. 片上网络映射问题的改进禁忌搜索算法[J]. 计算机辅助设计与图形学学报, 2008, 20(2): 155-160.
CHANG Zheng-wei, XIE Xiao-na, SANG Nan, et al. An Improved Tabu Search Algorithm for Network-on-Chip Mapping[J]. Journal of CAD & CG, 2008, 20(2), 155-160.

[7] GRECU C, JONES M. Performance evaluation and design trade-offs for network-on-chip interconnect architectures[J]. IEEE Transactions on Computers, 2005, 54(8): 1025-1040.

[8] CHANG K C, CHEN T F. Low-power algorithm for automatic topology generation for application-specific networks on chips[J]. IET Computers & Digital Techniques, 2008, 2(3): 239-249.

[9] GUERRIER P, GREINER A. A generic architecture for on-chip packet-switched interconnections[C]//Proceedings of the Conference on Design, Automation and Test in Europe. Washington, D C, USA: IEEE Compater Society, 2000: 250- 256.

[10] HOLLSTEIN T, GLESNER M. Advanced hardware/ software co-design on reconfigurable network-on-chip based hyper-platforms[J]. Computers & Electrical Engineering, 2007, 33(4): 310-319.

[11] KERNIGHAN B W, LIN S. An efficient heuristic procedure for partitioning graphs[J]. Bell System Technical Journal, 1970, 49(2): 291-307.

[12] CORMAN T H, LEISERSON C E, RIVEST R L. Introduction to algorithms[M]. Cambridge: MIT Press, 2001.

编　辑　蒋　晓