一种实时细颗粒度片上网络功率分配方法*

饶东宁，李建华†
（广东工业大学 计算机学院，广州 510006）

摘　要： 片上网络(NoC)不仅直接影响众核芯片的性能，而且消耗了芯片部分功率，在给定功率条件下使得片上网络的性能最优已经成为迫切需求。片上网络中路由器的工作负载(包的到达率)差异很大，需要对每一个路由器进行异构调整功率才能得到很好的性能。为此，针对如何在给定功率条件下优化 NoC 性能进行了研究，提出了一个实时细颗粒度的功率分配方法，它能够根据每一个路由器的工作负载实时快速地分配功率，实现片上网络性能最优。实验结果表明相比其他功率分配方法，该方法平均减少 26.1% 的应用执行时间且具有较低的硬件开销。

关键词： 众核芯片；片上网络；功率受限；性能最优；功率分配
中图分类号： TP301　　**文献标志码：** A　　**文章编号：** 1001-3695(2016)08-2358-05
doi:10.3969/j.issn.1001-3695.2016.08.027

~~Real-time~~ fine-gained power budget method for network-on-chip

Rao Dongning, Li Jianhua†
(*School of Computer*, *Guangdong University of Technology*, *Guangzhou* 510006, *China*)

Abstract: Network-on-chip (NoC) not only directly influences the overall many-core performance, but also consumes a significant portion of the total chip power. How to obtain the most optimal performance under the conditions of limited power has become an urgent need. The routers' workload (the packet arrival rate) in NoC is vary difference, so needs to adjust the power for each router heterogeneously to get good performance. Thus, this paper proposed a method of real-time fine-gained power budget in NoC under the limits of power which can ~~be on the basis of its~~ workload to ~~distribution~~ the power online and quickly to obtain the most optimize performance. Experimental results show that compared several state-of-the-art methods, the proposed method can reduce 28.9 percent of the application execution time averagely.

Key words: many-core chip; NoC; power-limited; optimal performance; power budget

0　引言

随着 CMOS 技术的发展，众核芯片广泛应用于云计算、移动高端设备和高实时性嵌入式终端设备等方面。但是技术的革新也带来了一些挑战[1]：伴随着众核芯片集成规模和整体性能不断提升，其功耗也快速增长。受到芯片封装、散热和外部供电能力的影响，众核芯片的功耗必须受到限制。这带来了众核芯片设计的挑战，即如何在给定功耗限制条件下使得众核芯片的性能最优。当前众核芯片设计的一个研究热点就是如何管理芯片功率，针对处理器，在已有的文献[2,3]中，采用动态调节频率/电压(DVFS)技术或者关闭闲置电路(power gating)技术对核进行功率调整，其基本思想是：当处理器核工作负载相对较低时就降低其电压/频率而当处理器核工作负载相对较高时就升高其电压/频率，或者关闭部分暂时闲置电路。然而片上网络(network on chip, NoC)是众核芯片处理器核之间进行数据通信的媒介，它的性能不仅直接影响着众核芯片的整体性能[4,5]，而且功耗大约占芯片总功耗的 10% －36%[6,7]。针对 NoC 的功率管理，需要考虑以下两个方面的因素：

a)每个路由器的工作负载是有差异的。图 1 显示了典型评测应用 BFS(breadth first search)[8]路由器工作负载分布。可以看出，20% 的路由器的包的到达率是 0 ~ 0.005 flits/cycle，而仅有 4% 的路由器的包的到达率为 0.03 flits/cycle 以上。如果为所有的路由器都分配同样的功率，那么工作负载较低的路由器可能分配过多的功率而工作负载较高的路由器功率不足够。

b)随着片上网络中路由器节点数目的增加，功率分配方法的状态复杂度呈指数级的增长。例如，假设 NoC 有 16 个路由器，每个路由器有四个可调频率，那么 NoC 就有 4^{16} 种频率组合。

文献[8 ~ 10]对 NoC 进行了功耗控制，根据其控制粒度，可以分为：a)芯片层控制，如文献[9]中提到的 PEPON 将整个 NoC 看做一个整体进行功率分配；b)粗颗粒度调节路由器功率，如文献[10]中提到的 VFIs，为了提高功率分配的粒度，将 NoC 中路由器划分区域，每个区域有相同的电压/频率；c)调节每一个路由器，如文献[11]中的 SAPP，每个路由器的频率是相同的。以上这些方法在管理 NoC 的功率方面都很有效，但是它们尚未考虑到每个路由器的工作负载的差异[9]，或者使得 Ad-hoc 方法[12]表现欠佳。实际上，优化 NoC 功率分配可以根据每个路由器的工作负载来调节其频率来实现。

鉴于此，本文针对 NoC 功率分配提出了一个在功率限制条件下，实时细颗粒度的功率分配方法 ReFiFS(real-time fine-grained frequency scaling)。它根据每一个路由器的工作负载实时快速细颗粒度的为其分配功率，使得 NoC 的总体性能达

收稿日期： 2015-07-17；**修回日期：** 2015-09-15　**基金项目：** 国家自然科学基金资助项目(61100134)；中央高校基本科研业务费专项资金资助项目(21615438)

作者简介： 饶东宁(1977-)，男，广东兴宁人，副教授，博士，主要研究方向为智能规划、嵌入式系统(raodn@gdut.edu.cn)；李建华(1989-)，男(通信作者)，~~硕士研究生~~，主要研究方向为嵌入式系统、智能规划.

到最优,达到最优能量管理的目的。本文主要贡献如下:a)在已有的性能分析模型基础上,建立了路由器工作频率和 NoC 性能之间的量化关系模型;b)针对如何在给定功率条件下最优芯片性能提出了一个实时细颗粒度功率分配方法,该方法能快速完成功率分配且具有较低的硬件开销。

1　模型和问题定义

1.1　NoC 动态功耗模型

在 NoC 中,路由器的功率与其电压和频率有关,所以构建路由器的功率和其频率之间的量化关系是有必要的。在文献[13]中,在给定路由器结构和电路实现情况下,可得到一个结构层的参数化 NoC 路由器功耗模型。例如,NoC 中有四个路由器,其功耗可以表示为:$\text{Power} = 2 \cdot f_1 + f_2 + f_3 + 2f_4$。则每个路由器的功耗可以表示为

$$p_i = g(f_i) \tag{1}$$

其中:$1 \leqslant i \leqslant 4$,$p_i$ 表示第 i 路由器的功率,f_i 表示第 i 个路由器的频率,g 是路由器工作频率和路由器功率之间的函数关系式。

1.2　NoC 性能模型

在 NoC 中,一个数据包由多个帧(flit)构成,第一个帧被称做帧头(head-flit),其他帧被称做数据帧(body-flit)。一个数据包的延迟包括帧头的等待时间和数据帧的传输时间。NoC 的性能可以用数据包的网络延迟表示,延迟低则表明 NoC 性能好。如 1.1 节中例子的性能可以表示为:$L_{\text{NoC}} = 1/f_1 + 2/f_2 + 4/f_3 + 3/f_4$,其中 $f_i \in \{2,4,6\}$。NoC 延迟可表示为

$$L = \sum_s \sum_d P^{s\to d} \times \left[\sum_{i=1}^{n} \varepsilon(i) \times [w_i + t_i]\right] \tag{2}$$

$$\varepsilon(i) = \begin{cases} 1, i \in \Phi_{s\to d} \\ 0, i \notin \Phi_{s\to d} \end{cases} \tag{3}$$

其中:s 表示源路由器,d 表示目的路由器,$P^{s\to d}$ 表示一个数据包从源路由器 s 产生并发送到目的路由器 d 的概率,n 表示网络中路由器数目,$\Phi_{s\to d}$ 表示某个数据包从 s 到 d 所经过的路由器集合,w_i 表示路由器 i 中 head-flit 的等待时间,b_i 表示路由器 i 中 body-flit 的传输时间。

由文献[14]可知,head-flit 的等待时间和 body-flit 的传输时间可以由式(4)和(5)表示。

$$w_i = \frac{\lambda_i}{2(1-\lambda_i/u_i)} \times E\left[\left(\frac{X_i}{f_i}\right)^2\right] + \frac{1}{2f_i} \times \frac{E(V^2)}{E(V)} \tag{4}$$

$$t_i = \frac{S_i}{W} \tag{5}$$

其中:λ_i 表示路由器 i 的帧到达率,μ_i 表示路由器 i 的服务率,X_i 表示路由器 i 的服务时间分布,f_i 表示路由器 i 的频率,V 表示路由器的平均剩余服务时间分布(当且仅当时钟周期有效的时候,一个帧的服务才有效,否则就需要等待),S_i 表示数据包的大小(依赖于应用特征),W 表示网络带宽。于是 NoC 中所有数据包的网络延迟可表示为

$$L = \sum_{i=1}^{n} [a_i \cdot h(f_i) + b_i] \tag{6}$$

其中:$a_i = \sum_s \sum_d P^{s\to d} \times \varepsilon(i)$,$h(f_i) = w_i$,$b_i = \sum_s \sum_d P^{s\to d} \times \varepsilon(i) \times t_i$,$a_i$ 和 b_i 独立于路由器的频率,可以周期性地计算得到。通过监测路由器帧的到达率等参数,式(6)可以周期性地得到更新。

1.3　问题定义

假设 NoC 有 n 个路由器,路由器的频率定义为 $f_1, f_2, \cdots, f_n$,每个路由器有 M 个可调频率 $F_1, F_2, \cdots, F_M$,即 $f_i \in \{F_1, F_2, \cdots, F_M\}$。假设路由器的功耗可以由式(1)表示,路由器的性能可以由式(6)表示。那么在给定功率 P 的限制条件下,最大化 NoC 的性能(NoC 网络延迟最小)问题可以转换成如下的问题:片上网络中每个路由器分别从 $F_1, F_2, \cdots, F_M$ 中选择一个合适的工作频率使得 NoC 的网络延迟最小(性能最优)且所有的路由器消耗的功率之和不能超过给定的功率 P。例如,1.1 与 1.2 节中的例子中假设功耗限制为 $P = 20$,那么在功耗限制条件下求取最优性能可以表示为:在 $2 \cdot f_1 + f_2 + f_3 + 2f_4 \leqslant 20$ 时求 $\min L_{\text{NoC}} = 1/f_1 + 2/f_2 + 4/f_3 + 3/f_4$,其中 $f_i \in \{2,4,6\}$。该问题在数学意义上就可以简单地表述为

$$\sum_{i=1}^{n} g(f_i) \leqslant P \tag{7}$$

$$\min L = \sum_{i=1}^{n} [a_i \cdot h(f_i) + b_i] \tag{8}$$

其中:$f_i \in \{F_1, F_2, \cdots, F_M\}$,$M$ 表示频率级别。

2　NoC 功率分配方法

2.1　方法总体设计

根据 1.3 节定义的问题,该问题类似于一个多选择的背包问题,n 个路由器对应于 n 组物品,第 i 组中有 m 个物品,对应于路由器 i 有 m 个可选频率,背包的容量即是 NoC 的可分配功率 P。一旦从某一组中选定一个物品装入背包,对应于某个路由器工作在某一个频率下。第 i 组中的物品 j 有唯一的价值 $V_{i,j}$ 和唯一的重量 $W_{i,j}$ 对应于在 NoC 中第 i 个路由器工作在频率 j 的时候有唯一的网络延迟值 $L_{i,j}$ 和唯一的功率消耗 $P_{i,j}$。

然而,背包问题可以利用动态规划算法在多项式时间内求得最优解,动态规划算法的求解过程实际上可以看做是一个求解多阶段网络的过程。鉴于此,1.3 节的问题也可以通过先构建网络然后采用动态规划的思想来求解。其求解思想如下:

设 $L_{i,p} = \sum_{j=1}^{i} [a_j \cdot h(f_j) + b_j]$,$(f_j \in \{F_1, F_2, \cdots, F_M\})$ 表示前 i 个路由器的总功率为 p 的条件下产生的最小网络延迟。$\sum_{j=1}^{i} g(f_j)$ 表示前 i 个路由器的功率之和。当第 i 个路由器设定频率为 f_k 时:

a)如果 $\sum_{j=1}^{i-1} g(f_j) + g(f_k) > p$,$L_{i,p} = L_{i-1,p}$

b)否则 $L_{i,p} = \min\{L_{i-1,p}, L_{i-1,p-g(f_i)} + a_i h(f_k) + b_i |_{f_k = F_m}\}$,$F_m \in \{F_1, F_2, \cdots, F_M\}$。

通过上面两步迭代 n 次后得到 $L_{n,P}$,即求得功率限制 P 下,n 个路由器的网络最小延迟。

为了能快速地求解上述问题,本文提出了一个 ReFiFS 方法,在该方法中,通过使用动态功率分配网络 PBN(power budget network,PBN)来实现实时功率计算和分配,其整体框架如图 2 所示。

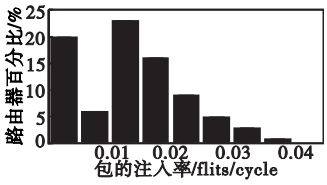

图1　针对评测应用BFS、NoC中路由器工作负载分布

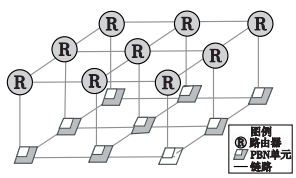

图2　支持动态功率分配的NoC整体框架

图 2 中每一个路由器配置一个 PBN 单元(主要负责在线功率计算和分配),所有的 PBN 单元构成一个 PBN。每一个

PBN 单元主要负责：

a)实时更新边的权重。在线监测 $P^{s\to d}$ 和 λ_i，PBN 单元中的计算单元按照式(8)实时计算网络延迟并更新本 PBN 单元和下一个 PBN 单元之间相应边权重。

b)求解 PBN 的最优解。NoC 功率分配问题可以使用 PBN 来解决，寻求路由器的最优频率等价于在 PBN 中遍历一条权重最短的路径。

c)在线完成功率分配。功率分配单元按照求解所得的频率值计算各自对应路由器的功率并进行功率分配。

在 PBN 中，每个节点表示不同的功率分配值。若路由器分配一个频率值产生的功耗等于相邻两个阶段中任意两个节点的功率分配值之差，则在这两点间加一条边。该边的权重是 $a_i h(f_i)+b_i$(该频率之下得到的网络延迟，由式(8)得到)。此时在 PBN 中遍历一条路径对应于一个在功率限制条件下可行的功率分配方案，遍历一条最短路径对应在功率限制条件下能得到性能最优(延迟最小)的功率分配方案。

最短路径方法求解该类问题最优解是十分有效的。如对 2.1 节的例子，若 NoC 平均分配功率，则频率只能取 $f_1=f_2=f_3=f_4=2$，得到的网络延迟是 5；若用分块管理功率方法，频率可为 $f_1=f_2=2, f_3=f_4=4$，得到的延迟是 3.25；但是若用最短路径方法求得频率为 $f_1=2, f_2=4, f_3=4, f_4=4$，得到的网络延迟是 2.5。

2.2　定义功率分配网络(PBN)

定义 1　功率分配网络(PBN)。PBN 是一个带权重的有向无环图 DAG(V,E,w)。在这个图中，V 代表图的节点集，E 代表边集，w 代表边的权重。

a)DAG(V,E,w)中有 $n+2$ 个阶段，用 $s_i(1\leqslant i\leqslant n)$ 表示，s_i 表示第 i 个 PBN 单元，即每个 PBN 单元包含 PBN 的一列节点，s_0 和 s_{n+1} 是虚拟阶段。

b)DAG(V,E,w)有 $(p_k+1)\times n+2$ 个节点，p_k 为不同的功率分配值，n 表示有 n 个路由器。图中每一列表示一个阶段，共有 $n+2$ 个阶段。其中阶段 0 和阶段 $n+1$ 是虚拟阶段。节点用 $v_{i,j}$ 表示，其含义是从阶段 0 到阶段 i 总功耗等于 j 的节点。

c)相邻两个阶段 s_i 和阶段 s_{i+1} 中的任意两个节点 $v_{i,j}$ 和 $v_{i+1,k}$，如果 $j+g(f_i)|_{f_i=F_m}=k$，$(0\leqslant i\leqslant n-1$ 且 $1\leqslant m\leqslant M)$，那么在节点 $v_{i,j}$ 和 $v_{i+1,k}$ 之间加一条边 $e_{(i,j),(i+1,k)}$。同时这条边的权重 $w_{(i,j),(i+1,k)}=a_{i+1}.h(f_{i+1})+b_{i+1}$，即表示路由器 $i+1$ 在此频率下产生的网络延迟为 $a_{i+1}.h(f_{i+1})+b_{i+1}$。

d)虚拟节点 S 和 D。阶段 0 中只有一个节点 S，用 $v_{0,0}$ 表示，阶段 n 中只有一个节点 D，用 v_{n+1,p_k} 表示。

图 3 中边上的权重是相应路由器在某个频率 f 下产生的网络延迟，若要得到 NoC 最小网络延迟，仅需要找一条从节点 S 到节点 D 的最短路径，同时记录下该路径上每条边上的频率，从而求得每个路由器的频率值，根据该频率值计算并分配每个路由器的功率。

当 NoC 给定功率 P 发生变化时，PBN 的结构不会发生变化(硬件设计时按照功耗模型式(7)已固化)，但是需要对网络的不同部分进行遍历求解最短路径，来更新一次所有的 PBN 单元的功率。例如在图 3 中，当功率从 p_k 变化至 p_{k-1} 时，仅仅需要对虚线 p_{k-1} 以上的点以及相连的边求解最短路径即可。NoC 功率限制类似于图中的虚线。通过将虚线向上或者向下移动来满足功率预算，虚线以上部分的最短路径即是问题的解。

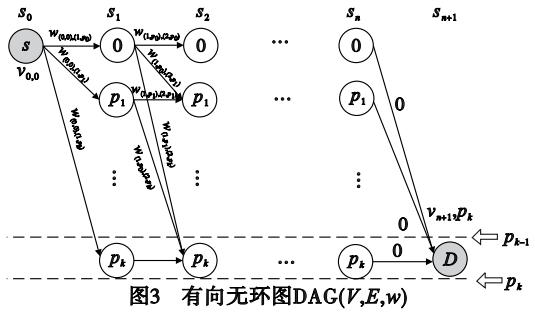

图3　有向无环图DAG(V,E,w)

2.3　更新 PBN 边的权重

由于网络中通信量有可能发生变化，就导致路由器工作负载(帧的到达率)发生变化，所以网络延迟模型中的第一阶段包的注入率 $P^{s\to d}$ 和第二阶段中路由器帧的到达率 λ_i 需要在线监测和计算，每个时间间隔更新 PBN 中每个边的权重 L。PBN 更新边的权重如算法 1 所示。

算法 1　PBN 更新边的权重

1. Input: $P^{s\to d}, \lambda_i$. $P^{s\to d}$: the probability of a packet is generated in node s and delivered to d. λ_i: the flit arrival rate of router i.
2. Output: $w_{(n,j),(n+1,k)}$: the weight of each edge after assigning f_i.
3. for each stage i from 0 to $n-1$ do:
4. for each node $v_{i,j}$ parallel do:
5. For each $e_{(i,j),(i+1,k)}$ connected to stage i and stage $i+1$ do:
6. if $P^{s\to d}$ or λ_i changed then
7. $w_{(i,j),(i+1,k)}=a_{i+1}.h(f_{i+1})+b_{i+1}$
8. return $w_{(n,j),(n+1,k)}$

在算法 1 中，共有 n 个阶段，每个阶段最多有 $P+1$ 个节点(P 为不同的功率分配值)，$P+1$ 个节点可在多个功率分配单元中并行计算。每个节点最多有 M 条边(M 表示频率级别个数，是常数且远远小于 n)，~~所以该算法的时间复杂度为 $O(n)$~~。

2.4　求解最优解

在得到 PBN 后，问题转换为在其中找到从 S 到 D 的最短路径。为了加快求解速度，本文利用最短路径并行算法从目的节点 D 到源节点 S 反向并行搜索。在一次迭代中，当前阶段的每一个节点选择一条边，这条边的权重和后一个阶段中与该边相连节点已得到的最小延迟之和最小，同时用该最小值更新当前阶段中该节点的延迟。在下一次迭代中，该最小值被传递给前一个阶段中的所有节点。每个阶段中的所有节点被划分到 PBN 中的计算单元同时计算，该搜索可以在 n 次迭代内完成。每个阶段的所有节点同时并行计算下面的值。

$$L_{\min}(v_{i,j})=\min_{\forall v_{i,j},\exists \text{an edge } e_{(i,j),(i+1,k)} \text{between } v_{i,j} \text{and } v_{i+1,k}}\{w_{(i,j),(i+1,k)}+L_{\min}(v_{i+1,k})\}\tag{8}$$

其中：$L_{\min}(v_{n+1,k})=0$，且 $w_{(n,j),(n+1,k)}=0$，即与虚拟节点 D 相连的边权重为 0。最终，计算可以表示为

$$L_{\min}=\min_{e_{(i,j),(i+1,k)}\in \mathrm{PATH}_{S,D}}\{\sum_{i=1}^{n}\sum_{0<j<p}\sum_{0<k<p}w_{(i,j),(i+1,k)}\}=\min\sum_{i=1}^{n}a_i\cdot h(f_i)+b_i\tag{9}$$

$\mathrm{PATH}_{S,D}$ 表示从原节点 S 到目的节点 D 的路径集合。那么在最优路径上的每一个节点 $v_{i,j}$，从 $v_{i,j}$ 到 $v_{i+1,j}$ 的最优路径(对应于最优的频率 f)可以由下面的等式得到

$$v_{i+1,k}=\underset{\forall v_{i+1,k},\exists \text{an edge } e_{(i,j),(i+1,k)} \text{between } v_{i,j} \text{and } v_{i+1,k}}{\operatorname{argmin}}\{w_{(i,j),(i+1,k)}+L_{\min}(v_{i+1,k})\}\tag{10}$$

算法 2　最短路径并行算法

1. Input: $w_{(i,j),(i+1,k)}$: weight of each edge.
2. Output: $L(v_{i,j})$: the minimum cycle of each node after assigning

f_i.

3. Initialize: All nodes $L(v_{i,j}) = \infty$, except $L_{\min}(D) = L_{\min}(v_{n+1,j}) = 0$/ and $w_{(n,j),(n+1,P)} = 0$

4. for each stage i from $n-1$ to 0 do:

5. for each node $v_{i,j}$ parallel do:

6. for each edge $e_{(i,j),(i+1,k)}$ connected to stage i and stage $i+1$ do:

7. if $L(v_{i+1,j}) + w_{(i,j),(i+1,k)} < L(v_{i,j})$ then

8. $L(v_{i,j}) = L(v_{i+1,j}) + w_{(i,j),(i+1,k)}$

9. $f_i = F_m(1 \leqslant i \leqslant n, 1 \leqslant m \leqslant M)$

10. return a short weight path from S to D and corresponding to the optimal solution.

并行求解最短路径如算法 2 所示。共有 n 个阶段,每个阶段最多有 $P+1$ 个节点(P 为不同的功率分配值),$P+1$ 个节点被划分到多个功率分配单元并行计算。每个节点最多有 M 条边(M 表示频率级别个数,M 是常数且远远小于 n)。每次迭代仅仅涉及到"加"和"比较"操作符~~,可以在一个周期内完成,所以该算法的时间复杂度为 $O(n)$~~。

3　实验

3.1　实验环境设置

实验验证仿真环境是两个基于 Alpha 架构的 CMP 和一个网络组成的 2D NoC 拓扑结构,每个 CMP 里面都集成了 Orion2.0[15]、McPAT[16] 和 CACTI 6.0 分别用来模拟 NoC、处理器和存储器的功耗,该仿真平台的详细参数配置如表 1 所示。

表 1　仿真器参数配置

仿真器组件	参数
处理器数目	128(基于 Alpha 架构的 64 位指令集)
取指/译码/提交大小	4/4/4
ROB 大小	64
L1 D 缓存(private)	16KB,2-way,32B line,2 cycle,2 ports, dual tags
L1 I 缓存(private)	32KB,2-way,64B line,2 cycle
L2 缓存(shared)	64KB slice,64B line,6 cycle,2 ports
主存储器大小	2GB
纳米技术	65nm
路由器数目	164
数据包大小	5 flits
flit 大小	72-bits
NoC 路由器延迟	2 cycle
NoC 链路延迟	2 cycle
NoC 缓冲通道	5×12 flits
路由算法	XY-routing

表 2 列出了三个典型评测应用,SSSP(single-source shortest path)[17]—在图中求解单源最短路径并行算法,BFS[8]—图宽度优先并行搜索算法,FastSort[18]—快速排序并行算法。

表 2　测评应用

测评应用	输入数据集
SSSP	(100 ~ 3000nodes)
BFS	(100 ~ 3000nodes)
FastSort	(20000 ~ 100000data)

3.2　NoC 延迟模型精确性分析

精确的性能模型在问题的解决中起到了重要作用。为了验证式(6)中的网络延迟模型的精确性,设置不同的包的注入率,比较仿真器模拟结果和延迟模型计算结果,利用式(11)计算结果如图 4 所示。

$$\varepsilon = \frac{L_m - L_s}{L_s} \times 100\% \tag{11}$$

其中:L_m 是模型延迟,L_s 是仿真器延迟。可以看出当包的注入率小于 0.04 flits/cycle 时,模型的相对误差小于 8%,所以,在包到达率较低的情况下,该延迟模型比较精确。

3.3　NoC 性能评估

在下面的实验里,通过将评测应用加载到仿真器上面运行,仿真器记录应用执行时间(正比于网络延迟),并将它作为一个量化指标来验证比较本文提出的功率分配方法和其他方法对系统性能的影响,应用执行时间归一化到本文提出的方法 ReFiFS。a) VFIs[10] 方法是将路由器分组,每组路由器同一个电压/频率。b) SAPP[11] 方法同构的调整路由器的频率。

在输入功率为 120W 条件下,当网络中通信量不同的时候(SSSP 输入图大小不同),评测应用 SSSP 归一化执行时间比较如图 5 所示。结果显示,相比 SAPP、VFIs 和 ReFiFS 分别平均减少了 27.6% 和 22.0% 的应用执行时间(最大分别减少了 32.4 % 和 28.6%)。

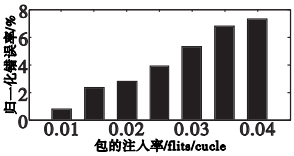

图4　性能模型平均错误率

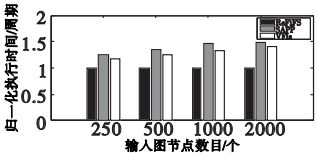

图5　输入功率为120W条件下,不同输入图大小,评测应用SSSP的归一化执行时间

当 NoC 输入功率从 120W 减少到 60W 时,BFS 归一化执行时间比较如图 6 所示。可见,相比 SAPP 和 VFIs,ReFiFS 平均分别减少了 30.1% 和 24.8% 的应用执行时间(最大分别减少了 35.4% 和 31%)。在相同输入功率条件下,三个不同评测应用 SSSP、BFS 和 FastSort 归一化执行时间如图 7 所示。可以看出在相同的输入功率条件下,针对不同的测评应用,相比 SAPP 和 VFIs,ReFiFS 平均分别减少了 29.9% 和 22.6%。

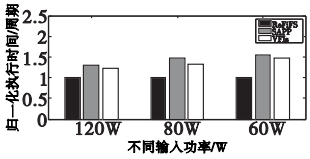

图6　当NoC输入功率从120W减少到60W时,评测程序BFS归一化执行时间比较

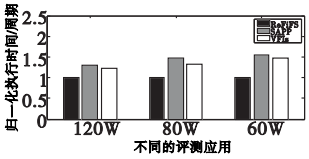

图7　在相同输入功率条件下,评测应用SSSP、BFS和FastSort执行时间比较

3.4　硬件开销

本方法的硬件开销主要是由于 PBN 中的节点,在 65 nm 下使用 Synopsys Design Compiler TSMC 库,得到每个节点的面积是 121 μm^2 且消耗 20μW 功率(假设 switching activity 是 0.5)。PBN 中总共有 $(p_k+1) \times n$ 个节点,实验平台一共有 164 个路由器节点,p_k 归一化为 10,那么总共需要的芯片面积是 218 284 μm^2,需要的功率大约为 36 mW。实验证明,在一个 8×8 的 2D 网络里面,PBN 仅仅占了很少一部分的芯片面积和功耗,硬件开销较低。

4　结束语

本文在给定功率条件下最优化 NoC 性能问题转换成求解最短路径问题。通过建立功率分配网络,在网络中求解最短路径,为每一个路由器找到合适的工作频率,达到了更优管理能量的目的。通过实验,相比于其他功率分配方法,在相同的输

入功率条件下,本文提出的方法能得到更少的应用执行时间,得到了最优性能。

参考文献:

[1] Venkatesh G, Sampson J, Goulding N, *et al*. Conservation cores: reducing the energy of mature computations[C]//Proc of ACM SIGARCH Computer Architecture News. 2010:205-218.

[2] Hanumaiah V, Vrudhula S, Chatha K S. Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors[J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems,2011,30(11):1677-1690.

[3] Ma Kai, Wang Xiaorui. PGCapping: exploiting power gating for power capping and core lifetime balancing in CMPs[C]//Proc of the 21st International Conference on Parallel Architectures and Compilation Techniques. [S. l.]:ACM Press,2012:13-22.

[4] Wang Chao, Yu Licheng, Liu Li, *et al*. Packet triggered prediction based task migration for network-on-chip[J]. Microprocessors and Microsystems,2014,38(4):316-324.

[5] Fu Weiwei, Yuan Mingmin, Chen Tianzhou, *et al*. Benefit of unbalanced traffic distribution for improving local optimization efficiency in network-on-chip[C]//Proc of the 6th IEEE International Symposium on Cyberspace Safety and Security, High Performance Computing and Communications. [S. l.]:IEEE Press,2014:76-83.

[6] Heck M J R, Bowers J E. Energy efficient and energy proportional optical interconnects for multi-core processors: driving the need for on-chip sources[J]. IEEE Journal of Selected Topics in Quantum Electronics,2014,20(4):332-343.

[7] Mattson T G, Riepen M, Lehnig T, *et al*. The 48-core SCC processor: the programmer's view[C]//Proc of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. [S. l.]:IEEE Computer Society,2010:1-11.

[8] Xia Yinglong, Prasanna V K. Topologically adaptive parallel breadth-first search on multicore processors[C] //Proc of IASTED. 2009:91.

[9] Sharifi A, Mishra A K, Srikantaiah S, *et al*. PEPON: performance-aware hierarchical power budgeting for NoC based multicores[C]//Proc of the 21st International Conference on Parallel Architectures and Compilation Techniques. [S. l.]:ACM Press,2012:65-74.

[10] Ogras U Y, Marculescu R, Choudhary P, *et al*. Voltage-frequency island partitioning for GALS-based networks-on-chip[C]//Proc of the 44th Annual Design Automation Conference. [S. l.]: ACM Press, 2007:110-115.

[11] Bhojwani P S, Lee J D, Mahapatra R N. SAPP: scalable and adaptable peak power management in NoCs[C]//Proc of ACM/IEEE International Symposium on Low Power Electronics and Design. [S. l.]: IEEE Press,2007:340-345.

[12] Chang K K W, Ausavarungnirun R, Fallin C, *et al*. HAT: heterogeneous adaptive throttling for on-chip networks[C] //Proc of the 24th IEEE International Symposium on Computer Architecture and High Performance Computing. [S. l.]:IEEE Press,2012:9-18.

[13] Kahng A B, Lin B, Samadi K. Improved on-chip router analytical power and area modeling[C] //Proc of Asia and South Pacific Design Automation Conference. [S. l.]:IEEE Press,2010:241-246.

[14] Li Shengnan, Pan Wenming. New latency model for dynamic frequency scaling on network-on-chip[J]. Journal of Electronic Science and Technology,2014,12(4):361-365.

[15] Kahng A B, Li Bin, Peh L S, *et al*. Orion 2.0: a power-area simulator for interconnection networks[J]. IEEE Trans on Very Large Scale Integration (VLSI) Systems,2011,20(1):191-196.

[16] Li Sheng, Ahn J H, Strong R D, *et al*. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures[C] //Proc of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. [S. l.]:IEEE Press,2009:469-480.

[17] 黄跃峰,钟耳顺. 多核平台并行单源最短路径算法[J]. 计算机工程,2012,38(3):1-3.

[18] 游佐勇,罗省贤. 多核计算环境下快速排序并行算法的实现[J]. 电脑与电信,2011(1):60-62.