基于螺旋线的 Round-Robin Crossbar 调度算法 [1]

张志群 魏激波 丁 炜 *

(国防科技大学电子科学与工程学院 长沙 410073)
*(北京邮电大学培训中心 北京 100876)

摘 要 该文提出了一种基于螺旋线的 Round-Robin(R-R) crossbar 调度算法，在调度级和迭代级分别轮询 R-R 指针，避免了指针同步；在输入端口轮询和迭代匹配的二维过程，执行流水线操作。当端口数 <32 时，执行效率高，带宽利用率高，逻辑简单，容易实现。通过对 R-R 加权，可以保证 Non-uniform traffic 高吞吐量。

关键词 VOQ, iSLIP, Crossbar 调度算法, Round-Robin(R-R), 流水线

中图号 TN919.3

1 前 言

随着链路速率的提高和端口数目的增加，由于受存储器带宽的限制，传统的共享存储、输出排队方式已无法适应核心路由交换的需要 [1−4]。基于输入排队、Crossbar 的交换技术因其交换容量大，受到广泛的关注，并在新一代核心路由器中得到应用。

排队机制和调度算法是影响交换规模和性能的两个主要因素。交换结构按照内部数据缓冲的位置和管理策略，分为输入、输出和中央排队 3 种模式 [5,6]。其中，输出排队性能好，最大吞吐量理论值为 1，交换矩阵调度简单，采用排队管理策略可以精确地控制 QoS 参数 (时延、带宽和公平性)，但要求 N 倍的加速因子，不适合大容量交换机。目前，大多数调度算法研究都集中在输出排队的 QoS 保证 [7]。中央排队的性能与输出排队相近，加速因子在 $[1, N]$ 之间，通过引入较复杂的 Buffer 管理逻辑，实现 Buffer 共享，实际上是牺牲 Buffer 管理和调度算法的复杂性来换取加速因子的改善，同样不适合大容量交换；输入排队加速因子为 1，但传统的 FIFO 输入排队存在著名的队头 (Head Of Line, HOL) 阻塞问题 [5,8]。这是因为：端口到达的所有报文都在同一个 FIFO 队列缓存，只有 HOL 报文才能得到调度，得不到调度的 HOL 报文会阻塞队列中发往其他空闲输出端口的报文，从而降低 Crossbar 的带宽利用率。Karol 研究表明 [9]：当报文到达服从 i.i.d. 的贝努利过程，N 趋于无穷大时，交换机最大吞吐量不超过 $2-\sqrt{2}=0.586$，对于突发过程，Crossbar 的带宽利用率更低。

然而，最近研究表明，基于虚拟输出排队 (Virtual Output Queuing, VOQ) 策略的输入排队能够解决 HOL 阻塞问题 [5,8,10]。VOQ 在输入端口对每个输出端口分别排队，消除了 HOL 阻塞，加速因子 =1，但是 Crossbar 交换开关调度复杂。因此，VOQ 获得高性能的关键在于设计高效的 Crossbar 调度算法，匹配输入队列和空闲的输出端口，即“二部图匹配”。

目前，很多研究已经集中在这个领域，并且 Tiny-Tera switch, cisco GSR 12000, BBN MGR 等核心路由器都采用了输入排队、Crossbar 交换开关结构。典型的单播最大匹配算法有 MBM(Maximum Bipartite Matching)[1] 和 PIM(Parallel Iterative Matching)[2]，MBM 算法复杂 ($O(N^{5/2})$)，并行差，硬件实现难，公平性差，会导致信元饿死现象。在 PIM 算法 [5] 中，通过 VOQ 方式可获得 100% 的吞吐率 [6,10,12]。

在 PIM 算法并行迭代的基础上，Nick McKeow 及其学生 Adisak Mekkittikul 分别在他们博士论文 [11,13] 中提出了 iSLIP(iteration SLIP), LQF(Longest Queue First), OCF(Oldest Cell

[1] 2001-11-15 收到，2002-04-22 改回
863 项目 (863-317-9601-02), Intel IXA 大学计划资助

First), OPF(Oldest Port First) 和 LPF(Longest Port First) 等算法. 尽管支持优先级调度的 EiSLIP 算法已经在 cisco GSR12000 系列路由器中使用, 但仍存在一些问题, 后面将详细讨论. LQF, OCF, OPF 和 LPF 等算法都是加权匹配算法, 用来提高 Non-uniform traffic 的吞吐量, 但存在的问题是逻辑复杂, 硬件实现难度大.

在流水线的思想上, 本文提出基于螺旋线的 Round-Robin(R-R) crossbar 调度算法, 它在输入端口串行调度和单端口迭代的二维过程中, 执行流水线操作, 它不要求每一次迭代都保证公平性, 而是在较长时间范围内保证公平性; 它不是通过端口并发匹配来保证每次调度的最大匹配数, 而是细化调整粒度和提高迭代的执行效率, 从而减少匹配时延, 保证吞吐量.

第 2 节扼要介绍了 VOQ 和 PIM, iSLIP 算法的原理; 第 3 节与 iSLIP 算法比较, 介绍了基于螺旋线的 R-R crossbar 调度算法的原理. 第 4 节对算法进行了仿真实验和性能分析. 第 5 节介绍了算法的实现. 最后, 总结了全文.

2 VOQ 机制和 PIM, iSLIP 算法

2.1 VOQ 机制

VOQ 基本模型如图 1 所示. 它由 N 个输入 / 输出端口、无阻塞的 Crossbar 交换机构和调度器组成. $A_{ij}(n)$ 对应从输入端口 i 到输出端口 j 的信元[2] 随机到达过程, $A_i(n) = \sum_{j=0}^{N-1} A_{ij}(n)$. 在输入端口 i 保持 N 个 FIFO 队列, $\mathrm{VOQ}_{i,1}, \mathrm{VOQ}_{i,2}, \cdots, \mathrm{VOQ}_{i,N}$. 同一 VOQ 中信元的输出端口一致. 若输入端口 i 到达一个目的端口为 j 的信元, $0 < i, j < N-1$, 那么该信元被放入相应的 VOQ_{ij} . 开关交换一个信元的时间为一个时间槽. 调度器决定每个时间槽内输入输出端口间的连接关系. 为了高效, 在第 n 个时间槽开关对信元交换的同时, 调度器根据 VOQ 状态确定第 $n+1$ 个时间槽开关的拓扑. 第 n 个时间槽结束, Crossbar 根据预先配置迅速改变内部拓扑, 并启动第 $n+1$ 个时间槽交换. 设在时刻 n, 队列 $\mathrm{VOQ}_{i,j}$ 长度为 $L_{i,j}(n)$, 那么, $L_{i,j}(n+1) = L_{i,j}(n) + a_{i,j}(n) - s_{i,j}(n)$, 其中

$$a_{i,j}(n) = \begin{cases} 1, & n\text{时刻有信元到达队列 } \mathrm{VOQ}_{i,j} \\ 0, & n\text{时刻无信元到达队列 } \mathrm{VOQ}_{i,j} \end{cases}$$

$$s_{i,j}(n) = \begin{cases} 1, & \text{第 } n \text{ 时间槽, 输入端口 } i \text{ 连接到输出端口} j \\ 0, & \text{第 } n \text{ 时间槽, 输入端口 } i \text{ 未连接到输出端口} j \end{cases}$$

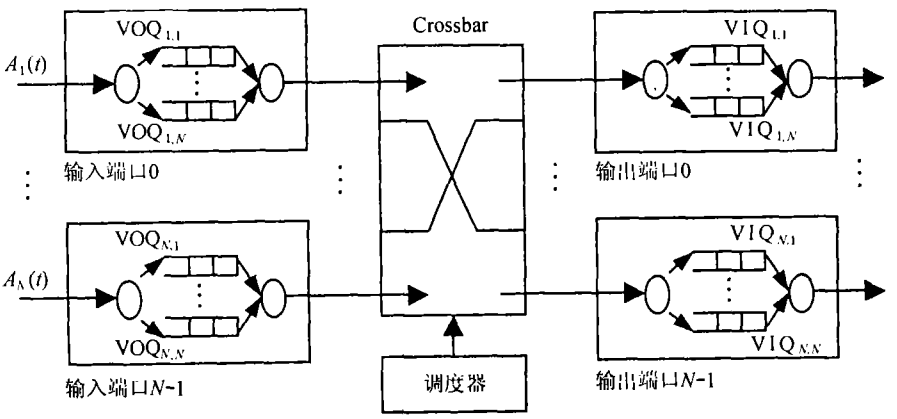

图 1　基本 VOQ 交换模型

[2] 由于核心交换为定长变换, 此处, 信元表示一个分组被分成 m 个固定长度的 PDU(Protocol Data Unit)

由于调度算法不能保证一个分组的 m 个信元在连续 m 个时间槽通过开关 ($m > 1$), 必须在输出端口设置 N 个虚拟输入队列 (Virtual Input Queue, VIQ), 每个 VIQ 对应一个输入端口, m 个信元重组成完整的分组再输出. 有关分组重组不在本文讨论 [14] .

2.2 PIM 和 iSLIP 算法

PIM 算法是经典的迭代最大匹配算法之一, 已在 DEC AN2 交换开关中应用. 每次调度由 2~5 次循环迭代组成, 每次迭代包括请求 (Request) 、授予 (Grant) 和确认 (Accept) 三个阶段:

步骤 1 请求 每个输入端口向其非空 VOQ 队列对应的输出端口发出接收请求.

步骤 2 授予 每个未匹配的输出端口根据收到的接收请求, 随机地选择一个输入端口授予, 并通知该输入端口.

步骤 3 确认 若未匹配的输入端口收到输出端口的授予, 随机地选择一个输出端口确认, 建立与该输出端口的连接.

PIM 算法通过随机匹配的方法来避免端口饥饿, 保证公平性, 减少实现最大匹配的迭代次数. 在一致贝努利分布情况下, PIM1 的吞吐率为 63%, PIM4 的吞吐率为 99% . 由于在授予和确认过程中使用了随机选择器, 当链路速率较高时, 硬件实现困难, 公平性差, 无法提供 QoS 保证.

iSLIP 也是三阶段迭代最大匹配算法. 它用 R-R 仲裁器取代了 PIM 的随机选择器, 由于使用 Round-robin 仲裁器, iSLIP 算法比 PIM 算法更容易硬件实现, 速度更快 [2,14,15] .

步骤 1 请求 每个输入端口向其非空 VOQ 队列对应的输出端口发出接收请求.

步骤 2 授予 未匹配的输出端按照 Round-robin 方式从具有最高优先级的输入端口开始轮询, 并且选择第一个要求的输入端口, 输出端口然后通知每个输入端口, 它的要求是否被授予.

步骤 3 确认 未匹配的输入端按照 R-R 方式从最高优先级的输出端口开始轮询, 确认第一个授予的输入端口. 这一步指针更新, 输出端口将输出 R-R 指针更新为匹配输入端口的下一个端口. 输入端口将输入 R-R 指针更新为匹配输出端口的下一个端口. 一次调度中, 每次迭代后未匹配端口进入下次迭代, 直至非空输入端口与输出端口最大匹配.

iSLIP 算法的优点是公平性好, 对于 N 端口的交换开关一次调度只需 $\log_2 N$ 次迭代, 易扩充支持组播和优先级调度等. 但是, McKeown 发现指针同步——指针指向同一的输入 / 输出端口, 会带来带宽利用率的下降, 在一次迭代中吞吐量可能不会超过 $1 - 1/e$=63% . 为了解决指针同步问题, iSLIP 通过指针互相参考来滑动指针 [15], 但是, 控制复杂, 硬件实现困难.

与其他最大匹配算法一样, 在 Non-uniform traffic 情况下, PIM 和 iSLIP 算法吞吐量都很低. 为了避免 Non-uniform traffic 情况下低吞吐量, 文献 [15] 提出 Threshold iSLIP 和 Weighted iSLIP 算法. 然而, 它们未提供在 Non-uniform traffic 情况下进一步性能分析.

3 基于螺旋线的 R-R Crossbar 调度算法

衡量 Crossbar 调度算法性能的主要指标有: 带宽利用率, 延时, 公平性, 复杂性. 影响 Crossbar 调度算法实现复杂性的主要因素有: 迭代次数, 并行度. 显然, 一次迭代匹配的端口数越少, 则设计越简单, 算法也越易流水实现.

从解决端口匹配的方式来看, 导致目前算法复杂的原因在于调度算法并行匹配所有端口, 所有输入 / 输出端口自由竞争所有输出 / 输入端口, 因此调度器内部不同端口的处理逻辑之间通信复杂, 并且不同端口的竞争和选择是同时进行的, 因此调度器需要复杂的仲裁逻辑.

从保证算法的公平性来看, 导致目前算法复杂性高的原因在于调度算法力求保证每次调度对每个输入输出端口都是公平的 [14] .

本文提出的基于螺旋线的 R-R crossbar 调度算法, 在输入端口轮询和单端口迭代的二维过程中, 执行流水线操作, 它不要求每次调度对每个端口都是公平的, 但算法在连续 k 次调度中

保证对每个端口是公平的，$k > 1$．算法采用两级 R-R 指针，在调度级，用 R-R 指针轮询各输入端口的请求，确定一个输入端口后；在迭代级，用 R-R 指针为输入端口选择一个输出端口．每次调度迭代一次，每次迭代只为一个输入端口选择输出端口，避免了 iSLIP 算法迭代时端口并行选择时而造成 R-R 指针同步引发性能下降．算法定义如下：

输入：第 k 次调度和请求向量 Req_i, $i = 0, 1, \cdots, N-1$, $\mathrm{Req}_i[j] = 1$ 表示 VOQ_{ij} 队列非空．

输出：输入 / 输出端口的匹配集合 $\Omega = \{< P_i, \mathrm{out} >, 0 \leq P_i, \mathrm{out} \leq N-1\}$．

其中，函数 $\mathrm{out}=\mathrm{MA}(\mathrm{Req}_i, R_i, \psi)$ 根据端口 P_i 的请求 Req_i，和 R-R 指针 R_i 以及空闲输出端口集合 ψ，为端口 P_i 选择一个输出端口．算法流程如图 2．

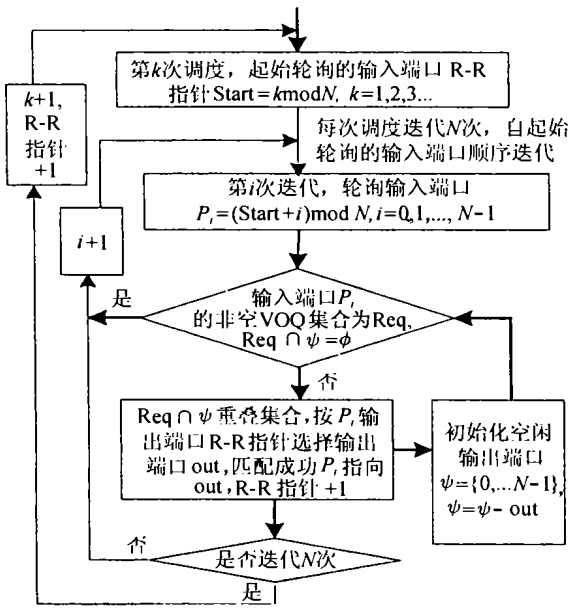

图 2　螺旋线 R-R 算法流程图

图 3 比较了螺旋线 R-R 算法与 iSLIP 算法的执行过程．螺旋线 R-R 算法执行螺旋线操作，在轮询输入端口和单端口迭代三阶段的两维方向，按照流水线模式组织起来．如图，输入端口 P_i 工作在请求阶段，P_{i-1} 工作在 R-R 仲裁阶段，P_{i-2} 工作在匹配阶段．实际上，一次迭代可看成 3 个端口并行工作，对 $N \times N$ 交换，N 端口调度要做 $N/3$ 次迭代．如表 1 所示，当 N 较小时 (核心路由器交换开关数目一般不大于 32)，螺旋线 R-R 算法不会成为交换开关的瓶颈．当 $N = 16$ 时，螺旋线 R-R 算法完成 N 端口调度的迭代次数略大于 iSLIP 算法，但可以通过减少单步迭代时间和提高输出端口的利用率来改善性能，第 4 节对此进行了仿真分析．

从逻辑复杂性来看，在 iSLIP 算法每次迭代产生 i 对匹配，$1 \leq i \leq N$，而螺旋线 R-R 算法每次迭代只产生 1 对匹配，因此调度器的 Crossbar 接口配置更简单．

螺旋线 R-R 算法可以理解为端口权值为 1 的加权匹配算法．可以对两级 R-R 加权，来保证 Non-uniform traffic 情况下的高吞吐量，实现优先级调度．由于算法按照二维流水线执行，可以将复杂的加权分散到调度级输入端口 R-R 指针和迭代级输出端口 R-R 指针．在调度级 R-R，根据优先级来轮询输入端口完成 N 端口调度，可以用 Longest port, Longest queue, Oldest port 或 Oldest cell 等加权因子作为输入端口权值，并且可以在第 k 次调度执行 N 次迭代的同时，

R-R 计算第 $k+1$ 次调度的输入端口加权因子；在迭代级 R-R, 可以用 Longest port, Longest queue, Oldest port 或 Oldest cell 等加权因子作为输出端口权值，并且可以在第 i 次迭代执行匹配的同时， $1 \le i \le N$, R-R 计算第 $i+1$ 次迭代的输出端口加权因子．整个过程逻辑不复杂，时延较小，运行时间为 $O(N\log N)$, 能够满足 Non-uniform traffic 高吞吐量的要求．而 iSLIP 等最大匹配算法要适合 Non-uniform traffic 要求，加权处理集中在 i 个端口并行执行，处理逻辑复杂，时延大，运行时间为 $O(N^3\log N)$.

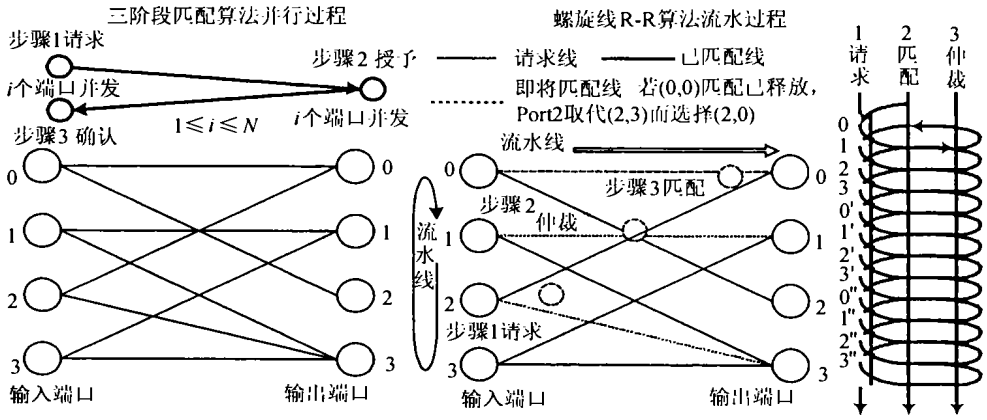

图 3 螺旋线 R-R 算法与 iSLIP 算法比较

表 1 完成 N 端口调度，算法迭代次数比较

	迭代次数 /N 端口调度	端口 =4	端口 =8	端口 =16	端口 =32
螺旋线 R-R 算法	$N/3$	1.34	2.67	5.34	10.67
iSLIP 算法	$\log_2 N$	2	3	4	5

4 仿真及性能分析

4.1 带宽利用率

在迭代中，不断有输入队列到达和匹配释放 (输出端口空闲), 未匹配图一直在变化． iSLIP 算法是并行迭代匹配，只有完成一次调度 ($\log_2 N$ 次迭代), 才能更新未匹配图，变化只能反映在下一次调度．螺旋线 R-R 算法调整粒度更细，完成一次迭代，更新一次未匹配图，匹配图的改变及时反映在下一次端口迭代，提高了输出端口利用率，从而提高吞吐量．图 4 表明了在信源到达服从不同参数的 ON/OFF 过程中，两种算法带宽利用率的仿真比较．

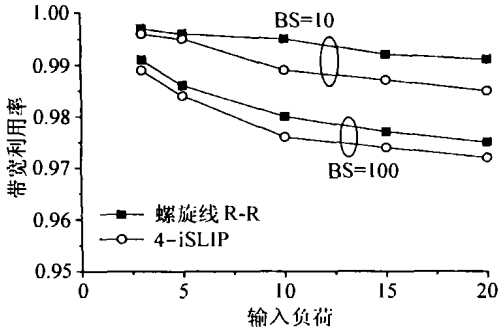

图 4　输入负荷与带宽利用率的关系，$N = 16$

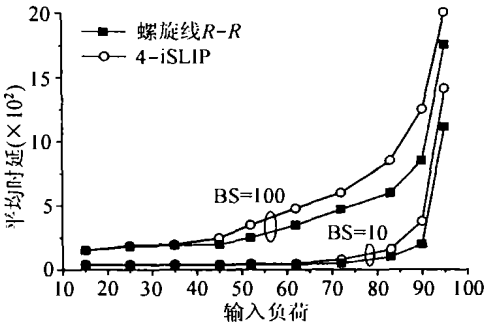

图 5　输入负荷与平均时延的关系，$N = 16$

轻负荷时，两种算法性能相近．许多 VOQ 队列都保持空状态，R-R 指针没有很大的作用，每个端口的调度相当于随机调度．算法在一次调度中匹配的端口数较少会导致下一次调度时非空 VOQ 队列数目的增加，从而下一次调度会匹配较多的端口．研究表明，当端口负荷较低时，所有使用 R-R 机制的调度算法都具有几乎相同的带宽利用率 [12]．

重负荷时，iSLIP 算法由于指针同步带来带宽利用率下降，性能略低于螺旋线 R-R 算法．并且，随着突发长度 (Burst Size, BS) 的增加，两者带宽利用率呈现下降．当 BS=10，螺旋线 R-R 算法带宽利用率一直保持在 99% 以上，当 BS=100，两种算法出现性能明显下降．

4.2 平均时延

在输入排队交换开关模型中，调度算法主要影响分组在 VOQ 中的等待时延．设时间区间 $[t_1, t_n]$ 内通过交换开关的分组记为 $c_1, c_2, \cdots, c_n$，分组 c_i 在 VOQ 中的等待时间为 d_i，$1 \le i \le N$．那么在区间 $[t_1, t_n]$ 内交换开关的平均时延为：$d(t_1, t_n) = \sum_{i=1}^{N} d_i/N$，$N$ 为开关端口数．

在 iSLIP 算法中，单步迭代有 i 个端口并发匹配，$1 \le i \le N$，需要并行授予和确认以及较长处理时间，处理逻辑复杂，单步迭代仲裁需 2 至 4 拍．螺旋线 R-R 算法借助流水线模型，将 iSLIP 的端口间请求、授予和确认的依赖关系 (一维) 转为单端口请求、仲裁和匹配三阶段和输入端口轮询的二维流水线操作，单步迭代仲裁需 1 拍，逻辑简单．即提高单步迭代匹配的执行效率来减少交换时延，保证吞吐量．

图 5 仿真表明，在 i.i.d. 或 ON/OFF 到达条件下，轻负荷时螺旋线 R-R 算法和 iSLIP 算法平均时延几乎相同，重负荷时螺旋线 R-R 算法的平均时延明显小于 iSLIP 算法．在两种算法中，信元的到达突发特性越强，信元的平均时延也就越长．

5 实　现

如图 6 所示，调度器由时序发生器、当前请求处理器、R-R 仲裁器和匹配 / 未匹配图组成．时序发生器产生时钟周期，1 个时钟周期为 1 个时间槽，对应请求和仲裁节拍，控制流水线操作步骤．当前请求处理器将输入队列分成不同的 VOQ 队列．R-R 仲裁器根据输入端口非空 VOQ 队列与空闲输出端口的交集选择输出端口．匹配 / 未匹配图用来建立匹配、释放匹配和生成空闲输出端口集合．

以实现端口速率为 10G，交换容量 160G，16×16 交换机为例．我们选用 Xilinx FPGA — VirtexEXC300，芯片工作频率为 200MHz，每个时钟周期为 5ns，算法完成一次调度需要 16 个时钟周期．同一端口相邻两次调度最差的情况是：在第 k 次调度首次被轮询的输入端口，在第 $k+1$ 次调度最后被访问，即要经过 31 个时钟周期；最好的情况是：在第 k 次调度第 2 次轮询的输入端口，在第 $k+1$ 次调度首先被访问，只需 15 个时钟周期．由于工作在流水线模式，同时进行 3 次迭代．所以，端口调度间隔时间 T，$(15/3) \times 5\text{ns}=25\text{ns} \le T \le (31/3) \times 5\text{ns}=51.7\text{ns}$．

信元的长度为 64byte, 为了实现交换开关的线速处理, 在 Work-conserving 情况下, 信元调度时间 $t \leq 64 \times 8/(10 \times 10^9) = 51.2\text{ns} \leq 51.7\text{ns}$, 加上线程上下文切换等开销, 此算法基本上可以在 FPGA 上实现端口速率为 10G, 16×16 交换开关的线速交换.

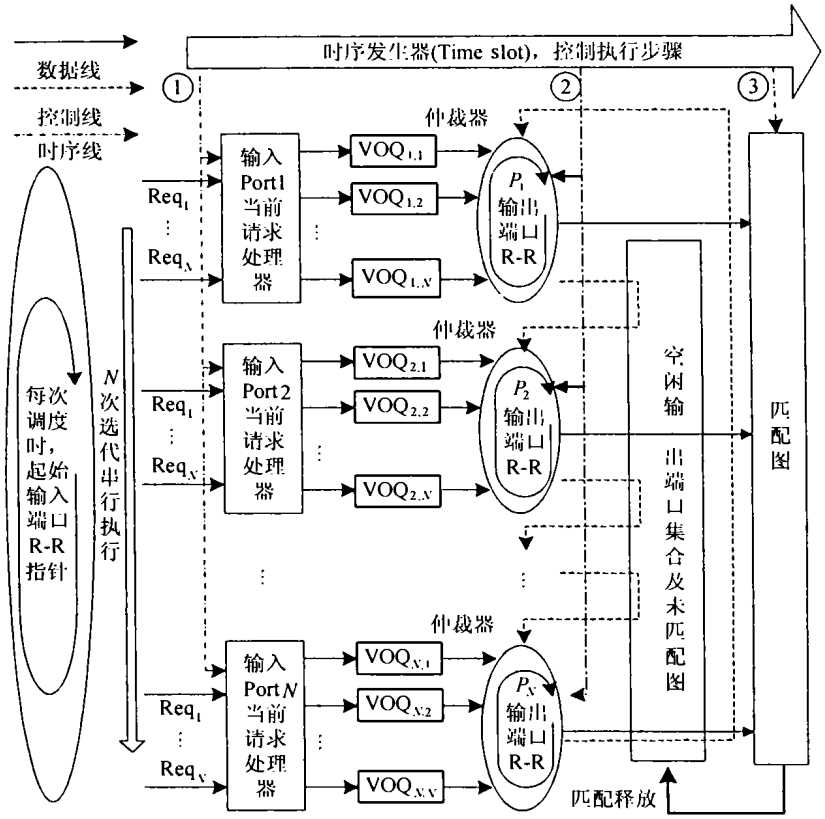

图 6 调度器实现框图

6 结 论

基于 VOQ 策略的 Crossbar 调度算法在核心路由器得到广泛的应用. 但包括 iSLIP 在内的大多数 Crossbar 调度算法都是直接基于 0.25μm 、甚至 0.18μm CMOS 技术实现的, 实现成本高, 所以允许算法控制逻辑复杂. 本文提出的基于螺旋线的 R-R 算法, 按照二维流水线的模式, 组织调度和迭代, 逻辑简单, 更易在 FPGA 实现. 同时, 能够保证高吞吐量和较小的交换时延. 通过两级 R-R 加权, 也可以很好地适应 Non-uniform traffic 的需要.

参 考 文 献

[1] S. Keshav, R. Sharma, Issues and trends in router design, IEEE Communications Mag., 1998, 36(3), 141–151.

[2] Nick McKeown, *et al.*, High performance switching (Proposal to Texas Instruments), Available at http://tiny_tera.Stanford.edu/~nickm/papers.html.

[3] Sundar Iyer, *et al.*, Analysis of a packet switch with memories running slower than the line-rate, Available at http://tiny_tera. Stanford.edu/~nickm/papers.html.

[4] Nick McKeown, Martin Izzard, The tiny tera: a packet switch core, IEEE Micro, 1997, 17(1), 26–33.

[5] T. Anderson, S. Owicki, J. Saxe, C. Thacker, High speed switch scheduling for local area networks, ACM Transaction on Computer Systems, 1993, 12(4), 319–352.

[6] Adisak Mekkittikul, Nick Mckeown, A practical scheduling algorithm to achieve 100% throughput in input-queued switches, Proceedings of IEEE Infocom'98, San Francisco, April 1998, 792–799.

[7] 王重钢，隆克平，等，分组交换网络中队列调度算法的研究及展望，2001, 29(4), 553–558.

[8] Y. Tamir, H. C. Chi, Symmetric crossbar arbiters for VLSI communication switches, IEEE Trans. on Parallel and Distributed Systems, 1993, 4(1), 13–27.

[9] M. Karol, M. Hluchyj, S. Morgan, Input versus output queuing on a space division switch, IEEE Trans. on Communications, 1987, COM-35(11), 1347–1356.

[10] N. McKeown, V. Anantharam, J. Walrand, Achieving 100% throughput in an input-queued switch, Proc. of INFOCOM, 1996, San Francisco, 296–302.

[11] N. Mckeown, Scheduling algorithms for input-queued switches, [PhD Thesis], University of California, 1995.

[12] Nick Mckeown, *et al.*, A starvation-free algorithm for achieving 100% throughput in an input-queued switch, Proc. of ICCCN'96, Maryland, Oct. 1996, 226–231.

[13] Adisak Mekkittikul, Scheduling non-uniform traffic high speed packet switches and routers, [PhD Thesis], Stanford University, 1998.

[14] 孙志刚，路由器高速交换开关调度算法的研究实现，[博士论文], 长沙，国防科技大学，2000, 10.

[15] P. Kupta, N. McKeowm, Design and implementation of a fast crossbar scheduler, Proc. of Hot Interconnects 6, Stanford, Aug. 1998, 77–84.

A R-R CROSSBAR SCHEDULING ALGORITHM BASED ON SPIRALITY

Zhang Zhiqun Wei Jibo Ding Wei*

(*National University of Defence Technology, Changsha* 410073, *China*)

*(*Beijing University of Posts and Telecommunications, Beijing* 100876, *China*)

Abstract In this paper, the Round-Robin crossbar scheduling algorithm based on spirality is proposed. It addresses the R-R pointers in scheduling and iteration, respectively, to avoid the synchronic of pointers. And it pipelines the polling of input ports and the steps of iteration in 2 dimensional. When the number of ports is less than 32, it has simple logic and performs better. In addition, by weighting the R-R, it can provide high throughput for non-uniform traffic.

Key words VOQ, iSLIP, Crossbar scheduling algorithm, Round-Robin, Pipeline

张志群： 男，1973 年生，博士生，讲师，研究方向：宽带 IP 网络.
魏激波： 男，1967 年生，教授，系副主任，研究方向：宽带通信网.
丁 炜： 男，1935 年生，教授，博士生导师，研究方向：宽带通信网.