**队列管理设计**

从工作场景的多样性以及产品成本的角度出发，许多研究者选择链表方案完成队列调度，本设计参考 PIEO，SP-PIFO设计了一种链表算法管理队列，它能以较为简洁的链表形式创建队列，完成出队、入队操作，有助于减少运行时因多次查表引入的延迟和后续的硬件设计。

本设计为每一个输入端口建立一个队列，一个队列管理一组共享缓存，一个队列中包含16x8条子队列，每条子队列记录着特定输出端口的特定优先级的包信息。

PQM模块的框图如下图所示：



PQM模块的端口信号的说明如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **端口名** | **方向** | **位宽(bit)** | **端口说明** |
| i\_clk | I | 1 | 时钟 |
| i\_rst | I | 1 | 复位 |
| i\_vld | I | 1 | 输入package的DA与Piror有效 |
| i\_da | I | 4 | package的目的端口 |
| i\_piror | I | 3 | package的优先级 |
| i\_addr\_in\_vld | I | 1 | 输入地址有效 |
| o\_addr\_in\_rdy | O | 1 | 可读入地址 |
| i\_addr\_in | I | SM\_AW | 输入地址 |
| i\_frame\_last | I | 1 | 输入一包的最后一个地址 |
| o\_addr\_rls\_vld | O | 1 | 回收地址有效 |
| o\_addr\_rls | O | SM\_AW | 回收地址 |
| i\_addr\_out\_rdy | I | 1 | 可输出地址 |
| o\_addr\_out\_vld | O | 1 | 输出地址有效 |
| o\_addr\_out | O | SM\_AW | 输出地址 |
| o\_out\_pending\_n | O | 1\*16 | 对16个目的端口的输出排队  （0<n<15,n表示输出端口号） |
| o\_out\_piror\_n | O | 3\*16 | 对16个目的端口的输出最高优先级  （0<n<15,n表示输出端口号） |
| i\_out\_vld | I | 1 | 输出许可 |
| i\_out\_port | I | 4 | 输出端口 |



算法模型如图所示，包括队列检索表和队列地址表，在队列检索表中存放每条子队列的队首地址和队尾地址，在队列地址表中存有若干地址节点，地址节点由数据和下一跳地址构成，一组数据块通过指针首尾相接，构成一条逻辑队列。出入队流程如下所示。

队列的管理内部有两级管理，第一级为队列检索表，存储着子队列的首尾地址，队列检索表存储在二维寄存器组内

|  |  |  |  |
| --- | --- | --- | --- |
| 队列号 | 队首地址 | 队尾地址 | 队列是否被创建 |
| 0 | head\_addr\_0 | tail\_addr\_0 | vld |
| 1 | head\_addr\_1 | tail\_addr\_1 | vld |
| ... | ... | ... | ... |
| 127 | head\_addr\_127 | tail\_addr\_127 | vld |

第二级为队列地址表，存储着子队列的所有地址，队列地址存储在SRAM内

|  |  |  |
| --- | --- | --- |
| 地址 | 下一跳地址 | 是否为帧尾 |
| 0 | next\_addr\_0 | frame\_last\_0 |
| 1 | next\_addr\_1 | frame\_last\_1 |
| ... | ... | ... |
| n | next\_addr\_n | next\_addr\_n |