交换模型switch基类函数列表

|  |  |
| --- | --- |
| RapidIO交换系统文件简介 | |
| base\_routable.h | 路由表基类头文件 |
| base\_router.h | switch基类头文件，内附基类实现 |
| centralized\_buffer.h | 在systemc中原文件的基础上，对选择弹出一个数据包函数稍微修改 |
| circular\_buffer.h | 环形buffer头文件，在systemc的基础上增加了push\_sort函数 |
| common.h | payload结构体定义及需要的库文件引入 |
| dispatch.h | switch buffer调度模块 |
| init\_adapter.h | 发送者到switch的适配模块 |
| initiator.h | 发起者模块 |
| routable.h | RapidIO路由表模块头文件 |
| router.h | rapidIO switch模块头文件 |
| soclib\_payload\_extension.h | payload拓展数据部分头文件及实现 |
| targ\_adapter.h | switch到接收者适配模块 |
| target.h | 接收者模块 |
| centralized\_buffer.cpp |  |
| circular\_buffer.cpp |  |
| initiator.cpp |  |
| router.cpp |  |
| target.cpp |  |
| top.cpp | 系统最上层文件 |
| Makefile | makefile文件 |
| README.md | 各个版本迭代信息简记 |

上面表格记录了RapidIO交换系统涉及的主要头文件及其实现文件，主要有发送方模块，发送方适配器，接收方，接收方适配器，路由表，switch，包调度模块以及两个基类模块。

其中circular\_buffer.cpp中新增了push\_sort函数，该函数用于对接收方接收到的数据包按照扩展中包的pkt\_id进行排序模拟1x模式，4x模式按时间顺序进行调度转发，默认情况下就是按照时间顺序进行弹出数据包转发。

Centralized\_buffer.cpp中修改了pop函数，原来的pop函数对空buffer也参与竞争调度和设置时间，而修改后的pop函数对空buffer不参与竞争。

Router\_table.txt文本文件中静态建立RapidIO路由表时的初始路由表，后期可能改为通过jtag方式建立静态路由表。

Readme文件简单记录各个版本新添加的信息，对于删除修改的信息则未记录。

Makefile文件进行相应修改，主要是添加了相应.cpp文件，当需要考虑虚拟网络与真实网络互联时，则需要在makefile文件中g++后加入-pthread参数。将代码集成到38所提供的代码时，需要代码进行微调同时修改所里代码的makefile文件，加入对soclib\_payload\_extension库文件的包含。如果加入网络模拟多线程代码，则还需要加入-pthread参数。

设计init\_adapter和targ\_adapter是由于switch采用tlm2.0的simple\_socket，而所里代码模拟的是tlm\_socket，所以设计了适配器用于tlm\_socket和simple\_socket的相互转化。Switch选用simple\_socket是为了标识数据包从哪个端口进来，并根据此设计相应的处理逻辑和进入哪个switch buffer。设计以太网switch时，需要保留特别的端口用于通过socket网络编程实现客户端与服务器端TCP连接，进行数据交互。

Initiator文件模拟发送方，在top文件中实例化initiator类时，将其中一个的设备ID设为0代表host，该设备通过check函数检查自己设备ID为0后，执行系统初始化枚据，建立整个系统的拓扑结构。在系统初始化过程中，host发送的都是维护数据包，并且设置targetid为0xff代表初始化维护包，然后根据switch和endpoint的响应，再发送维护写数据包设置switch路由表路由项。

Target文件模拟接收方，接收方设置对接收到的报时否进行目的ID校验，并且发送响应数据包，发送的响应数据包还是接受到的数据包的引用，所以对响应数据包的设置就是对接收到的数据包的设置。考虑两种处理方式，一是将接收方收到的数据包复制一份进行处理，将原来的修改后作为响应包返回给发送方；二是将收到的包复制一份作为响应包返回给发送方，对收到的数据包进行处理。

Router文件中主要完成对到达数据包的存储转发，根据输入端口送入对应的buffer中，然后在每个时钟周期选出一个数据包，将数据包路由到目的地。通过接收目标发回的响应数据包，对响应数据包采取类似直通转发方式返回到发送方。