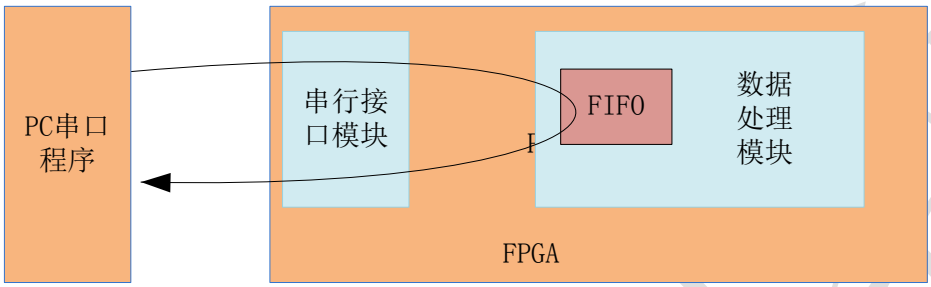
串口指令控制 FIFO 环回功能练习思路

一、 练习内容及要求

本练习实现串口环回功能，并通过练习，认识上位机和 FPGA 交互指令的使用。



其具体功能如下：

- FPGA 内部有一个可保存 128 字节的 FIFO；
- 上位机按以下格式发送指令：1 字节的类型(type)+1 字节的数据(data)。注意，每次发送必须是两个字节一起发送。
- 如果 type 为 0，则将 data 写到 FIFO 中。如果 type 为 1，则后面的 data 无效，同时模块将 FIFO 保存的个数返回给上位机。如果 type 为 2，则后面的 data 无效，同时读取 FIFO 内 1 个数据，并返回给上位机，如果 FIFO 本身没有数据，则不返回。如果 type 为其他，则 data 无效，同时模块无任何操作。

该串行接口的参数如下：

波特率：9600；无校验位。数据位：8 位；停止位：1 位；按十六进制发送接收。

二、 设计思路

1. 指令识别

本练习约定 2 个字节共同组成 1 个指令，通过 type 区来进行指令识别。由于数据是字节单位来传输的，type 和 data 域交替进来，可以用信号 rec_flag 来进行区分。当 rec_flag 为 0 时，进来的数据是 type；当 rec_flag 为 1 时，进来的数据是 data。

同时，rec_flag 也是计数器，用来对指令长度计数。初值为 0，加 1 条件为接收到一个数据，结束是 1。

指令的 type 部分要保存，等于指令接收完整时进行识别。假设保存的信号是 type，则当 rec_flag==0&&din_vld 时，type <=din;

2. 写 FIFO

当 type 为 0 时写 FIFO，因此 wrreq 为 1 的条件是：rec_flag && din_vld && type==0。写的值就是 din。

3. 发送个数

当 type 为 1 时将 FIFO 的个数发送，即：rec_flag && din_vld && tppe==1 时发送，此时发送的数据是 usedw。

4. 读 FIFO

当 type 为 2 时读 FIFO，因此 rdreq 为 1 的条件是：rec_flag && din_vld && type==2，并同时将 q 的数据发送给接口模块。

5. 交互指令认识

上位机与 FPGA 之间的控制，一般都通过本练习中的类似指令来完成的。本练习的指令虽然是最简单的，但具有一般性形式。

- 通常指令由多部分组成，每部分都有特定的意义，例如本例中的 type 和 data。
- 每条指令通过特定意义的不同值为进行区分。
- 每部分的长度可固定，或者能指明一定长度。本例中都固定为 1 字节。也可以通过某部分来指定长度。例如类型+长度+数据的指令。
- 指令尽可能让 FPGA 处理简单，一步到位。不要让 FPGA 做复杂的、多步的操作。这是因为上位机复杂控制非常简单，而 FPGA 则非常麻烦。例如，假设一条指令要 FPGA 先后做 ABCD 四个步骤功能，还不如分成四条指令分别对应 ABCD 四个步骤，FPGA 收到哪条指令就执行哪个步骤，由上位机来保证顺序正确性。
- 由上面 2~4 各指令实现可知，各个指令之间是独立的。但有部分同学考虑得更仔细，假如这些指令有冲突怎么办？例如读 FIFO 指令还没处理完，又来一个读 FIFO 指令如何处理？可选的方法包括：
  - 等待前面的处理完，再处理新指令。这样就需要先保存新指令。但问题又来了，恶劣情况下连续来好几条指令，那要保存多少呢？理论上永远也保存不够。更严重的是，FPGA 就做得非常复杂了，考虑的异常情况很多，而这并不是 FPGA 擅长的。
  - 直接丢弃处理。这种处理方式很简单，主要是功能上能否接受。如果能接受，则是最好的处理方式。但能接受的情况估计很少。
  - 上位机等待。做法是上位机发送一条指令后，延时一段时间或等待某个信号，确保指令执行

完毕，再发送下一条指令。明德扬推荐这种方法，暨能够实现简单、保证通信效率，又不会丢失指令。

本练习无须担心这个问题，是因为上位机发送一条指令，即两个字节，FPGA 才返回一字节数据，无论上位机发送指令有多快，FPGA 都能够处理，不会出现多个指令冲突情况。