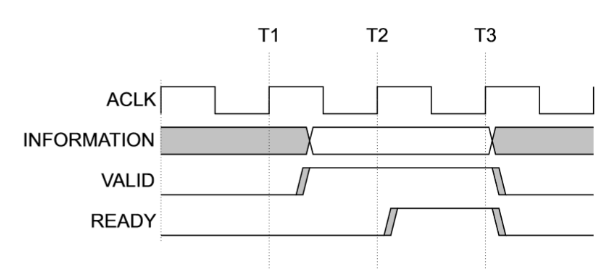
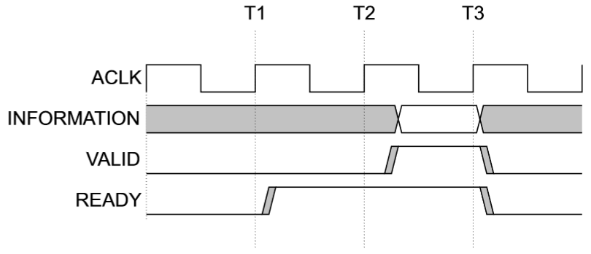
书写工程时，一定要了解好是时序的规则

分别为**VALID 信号先到达，READY 信号先到达，同时到达**

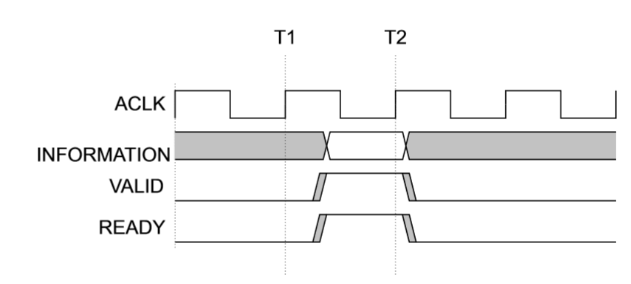
**VALID 信号先到达：**



**READY 信号先到达：**



**同时到达：**



问题总结：

在上一次的设计中，没有考虑到vaild信号产生的条件没有考虑。拉高的时候，只有ready拉高了才能置低。所以在仿真的时候就应该避免，vaild拉高的时候，没有产生ready，这个操作是不对的

以上的问题由仿真解决，因为在master中，做了为了等待ready信号为1的话，保持vaild的信号，这样导致时序对不上，用组合逻辑即可。

首先，一定要清楚，一旦vaild拉高了，就一定会传输数据，一定要等到ready信号拉高了，才能等拉低了。

Forward：

核心问题：

首先，forward会在payload和vaild，之间加入寄存器，为了让时序收敛。

但是这里有一个问题，当payload和vaild，发给下一级寄存器的时候，我们的我们需要等待多一个时钟周期，才知道下一步的操作。

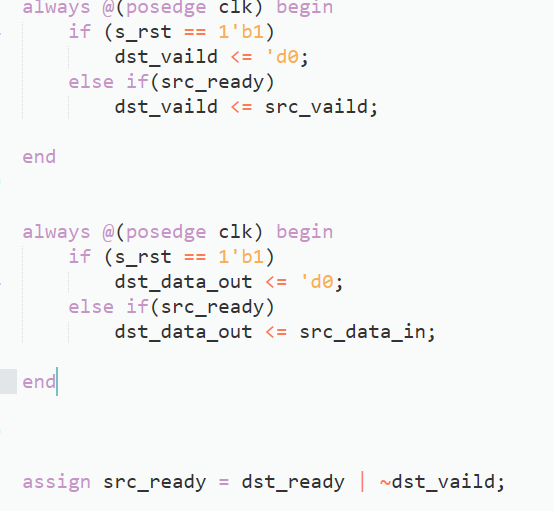
Ready反馈不及时，导致数据多保持一个周期，多次采集第一数据。如下图



核心概念：想要流水，则需要把中间的payload和vaild的寄存器当作master，将真的master隔离开。这样假master和slave做流水，然后真master配合假master即可。

解决方案：

需要让payload和vaild的寄存器当作master，当payload和vaild的寄存器没暂存数据的时候，我们应当做一个假ready，让他拉高，所以则是，假ready的信号里面，有一个！vaild\_out,这样当发起vaild的时候，则可以第一下，寄存器可以采集到payload和vaild。这样我们假master就能当作真master了。可是我们如何跟slave递交数据呢，这时候，我们就需要只靠考虑假master与slave，利用假ready信号，选择写入slave。



！Dst\_vaild信号既暂存payload数据和vaild数据，将假master和slave连接起来，通过反馈的src\_ready,也把数据锁了起来，或者是给选择的数据

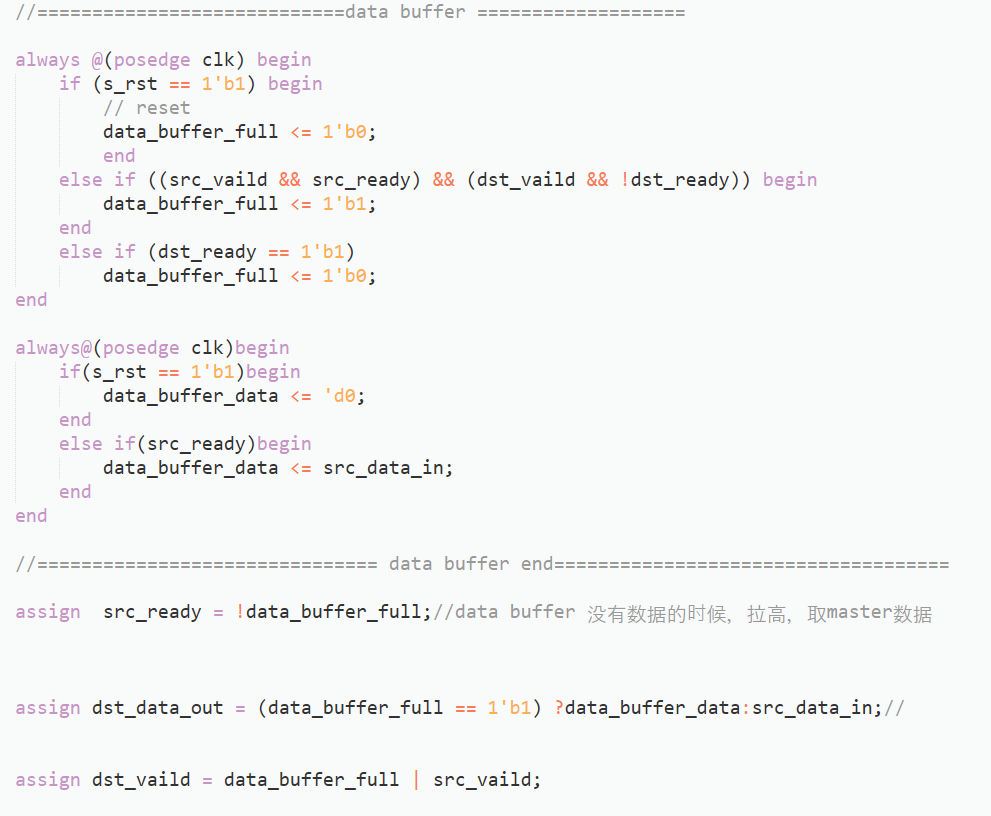
Backward，

核心问题：当我们，进行payload和vaild有效的时候，ready会滞后一拍子，导致数据多采集一次。数据也不得不多保持一拍。

核心概念：需要有一个data\_buffer缓冲器，可以提前收集一个数据。分离开master和slave。通过data\_buffer缓冲器，输出vaild的，取反递给master的ready，从而实现了ready的插入寄存器。

Data\_buffer,将master和slave分隔开，buffer没有数据时一直向ready拉高数据，有数据拉低。

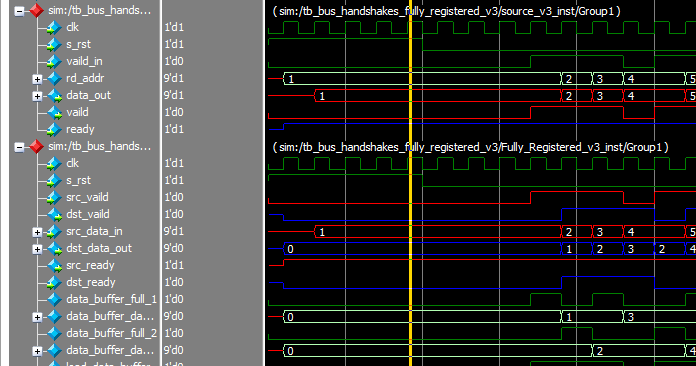
但是当slave的ready信号很多的时候，同时master的vaild信号拉高，data\_buffer数据不够存，则是选通了master的payload和vaild的从而实现，master和slave的直通，实现流水。



Full：

核心问题：在vaild拉高的时候，至少需要4个周期，才能发送下一个payload。

提前将ready的信号拉高，需要databuffer存，但是又因为master和slave之间插入寄存器，又必须要流水。在存到databuffer的时候，又来一个payload需要存，（如果是Backward操作，那就不再buffer之间将数据直接递出去即可，因为fully）。



核心和方案：我们再加了一个datebuffer，进行数据存储。这样就能做到两种方式的数据，都可以通过寄存器写入slave，实现fully。