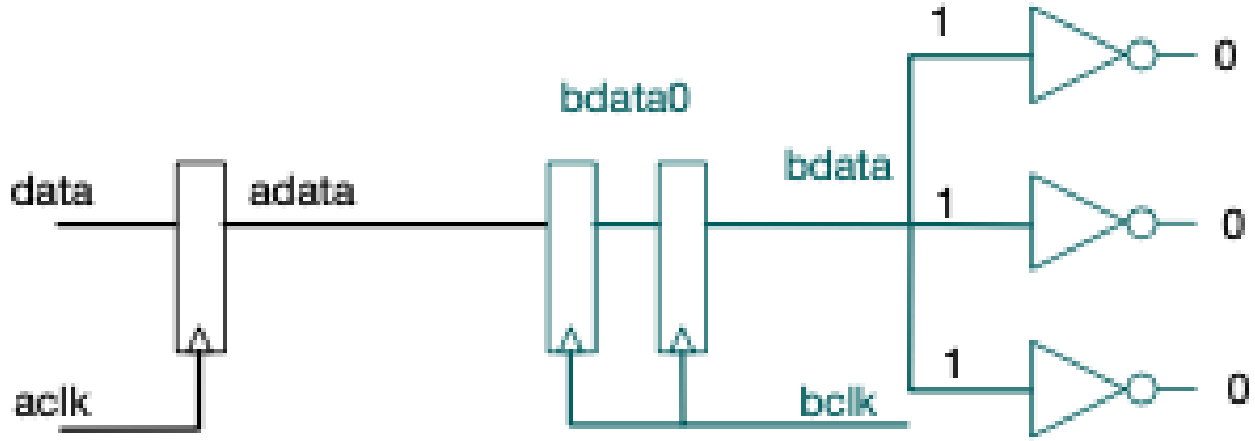
你真的懂2-flop synchronizer吗-- CDC的那些事（2）

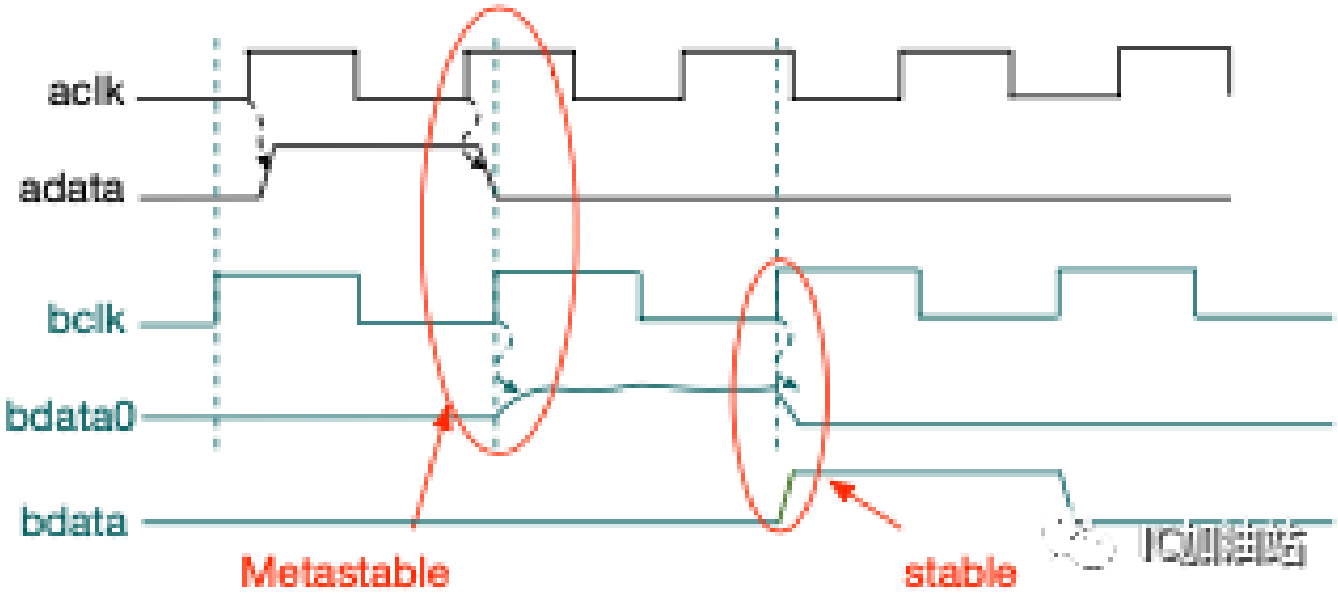
原创 老李飞刀 IC加油站 5月26日

这是CDC系列文章的第2篇。第一篇链接在
跟老李一起学习芯片设计-- CDC的那些事（1）

上一篇中我们回顾了一些基础知识，其中最重要的概念就是亚稳态。我们接下来所要看到的各种CDC的设计方法，本质上都是围绕在如何解决亚稳态带来的问题。

我们首先来看最基本的问题，single bit level 信号的跨时钟域。single bit 直接被destination domain的flop去sample产生的问题我们在上一篇已经讨论过，那么解决的办法呢？看起来很简单 -- 之后再加一个flop，也就是说用两级的flop来同步source domain 的 signal 。 我 们 通 常 把 这 种 synchronizer 叫 做 2flop synchronizer 或 者 double flop synchronizer，俗称“打两拍”。





说实话，老李在最开始学习到这个办法的时候，内心的声音是：“这TM在逗我？这么简单就可以了吗？凭什么第二级的输出就没有亚稳态了？”相信有很多初学者也和我当初有同样的困惑。在这里我们要再次回顾一下metastable产生的原因。第一级flop产生metastable的原因是flop里面没有及时锁住该锁的值，所以我们无法直接使用第一级flop的Q来直接用于bclk时钟域。但是要注意，我们之前说过，第一级flop的Q**会最终稳定下来**的，而且在**绝大多数时候**，可以在一个bclk周期内稳定下来，这样第二级flop的D输入就是一个稳定的值，进而第二级flop的Q是满足clk-to-q的，没有亚稳态的产生。如上图所示，尽管bdata0产生了metastable，但是bdata是stable的。

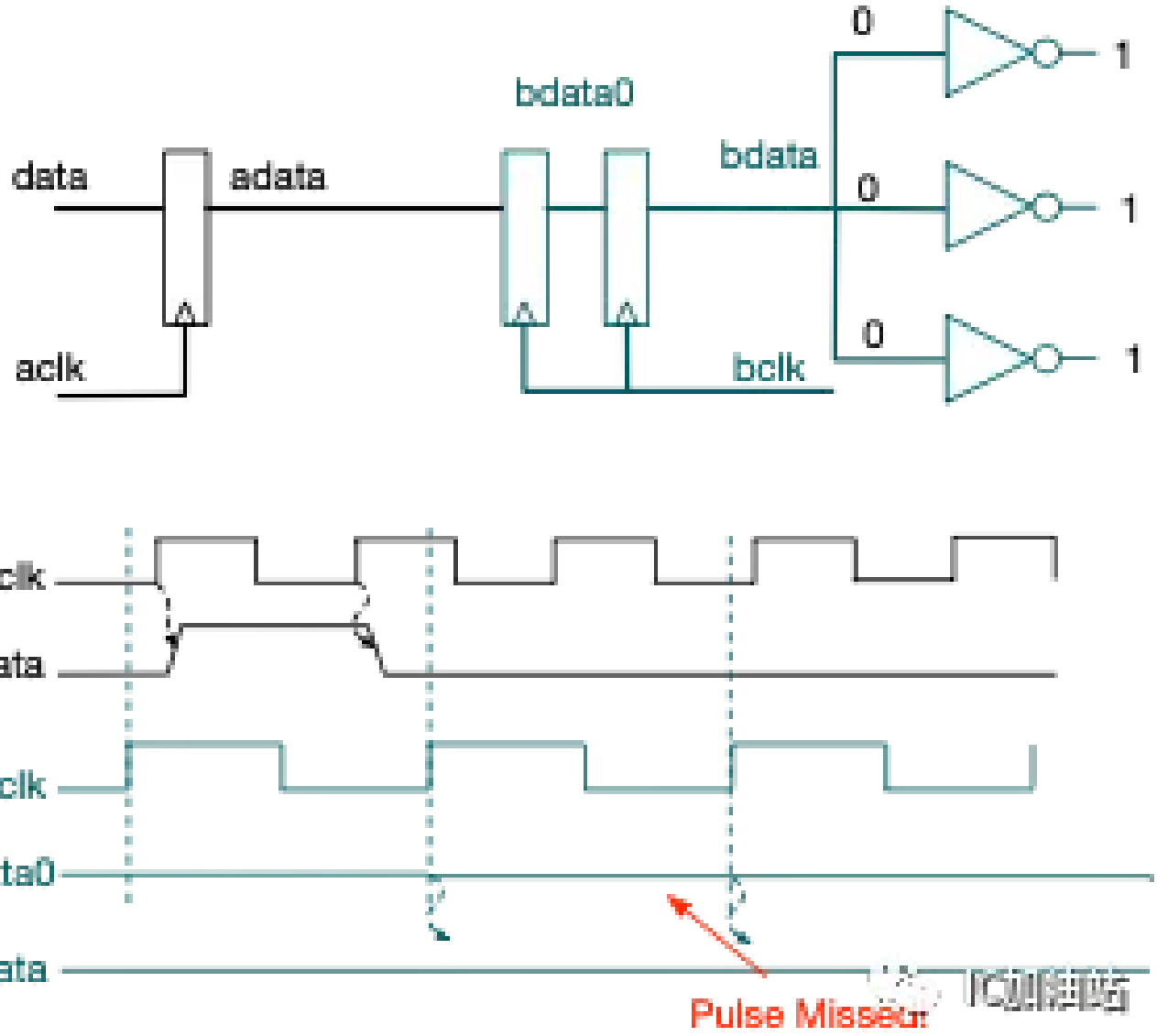
那你可能会问，上一讲你不是说第一级的flop的输出可能需要很长时间才能稳定吗？你凭什么说在第二个时钟沿到来的时候bdata0就能稳定呢？如果还是不稳定，让第二级flop产生了setup/hold time violation，那第二级flop不还是有可能产生metastable么？

这是一个非常正确的问题，的确，double flop synchronizer不能完全消除亚稳态！但是很多有经验的工程师会告诉你，用个double flop synchronizer就够了，那是因为double flop会使得metastable产生的概率显著降低，这就又回到了我们上一讲的MTBF的概念。在使用double flop的时候，由于给了第一级flop一个周期的时间去稳定，使得两级发生metastable的概率大大降低。我知道大家都不喜欢看公式，下面给了一个100MHz时钟的例子，大家只需要关心一下最终的结果，是957亿年，而地球的年龄是46亿年，太阳的寿命是100亿年，也就是说，直到太阳毁灭，你也碰不到下一次metastable的产生。

$$MTBF(t_r) = \frac{e^{t_r/\tau}}{T_0 \cdot f_{clk} \cdot f_{data}} \cdot \frac{e^{t_r/\tau}}{T_0 \cdot f_{clk}} = 9.57 \times 10^{10} years$$

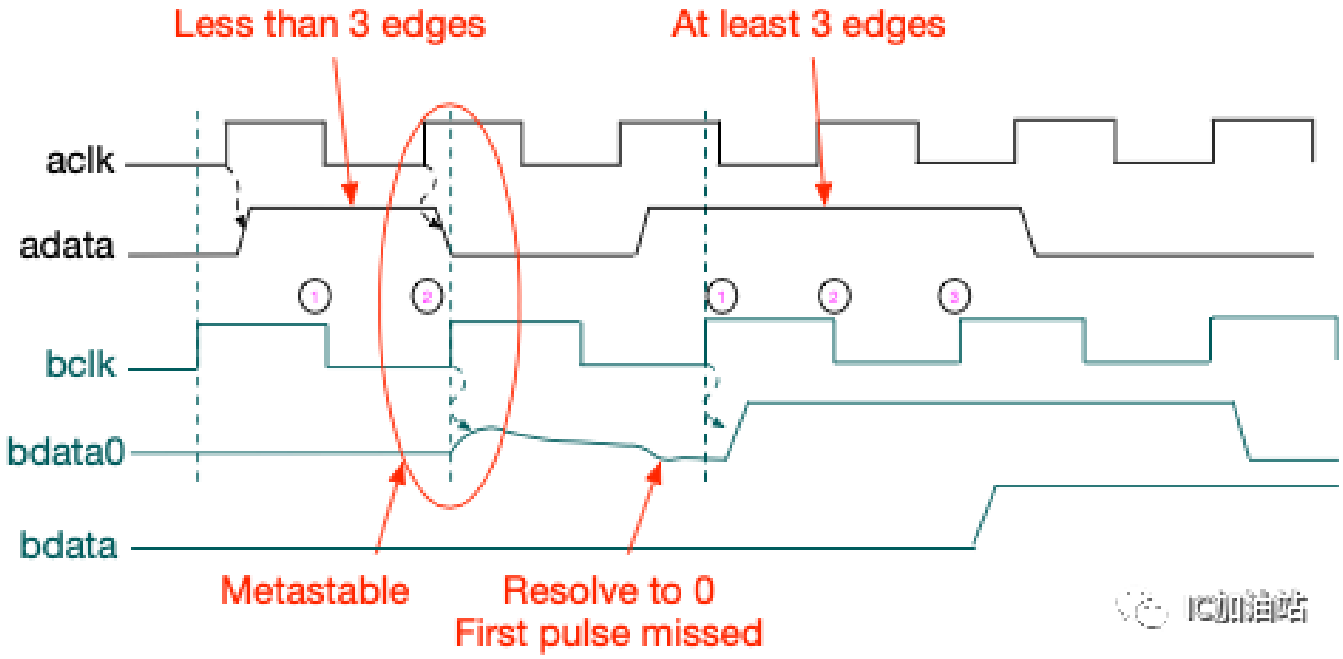
但是注意，随着sample clock frequency的提升，以及工艺节点越来越小，有些时候打两拍已经不太够了，那么就简单粗暴来一个打三拍，就能够保证了。而老李目前还没有看到过有谁在设计中实现打四拍的，如果你见过，请联系我，老李更愿意和他聊聊地球毁灭和人类未来的话题。

想当初，老李刚刚学到一点double flop synchronizer，以为CDC就不过如此，可以仰天大笑出门去也。但是之后的一个fail test就把当时还是小李的我给搞懵了。打开波形一看，搞什么鬼，信号没有同步过来啊？



不是说好了单比特信号用double flop么，肯定是哪里不对？老李仔细一看，这adata怎么在bclk的沿到来之前又变了呢，这样bdata0这个flop压根看不到adata的变化啊，看来double flop来做synchronizer是有条件的，不是随便什么单bit信号都可以直接无脑用！

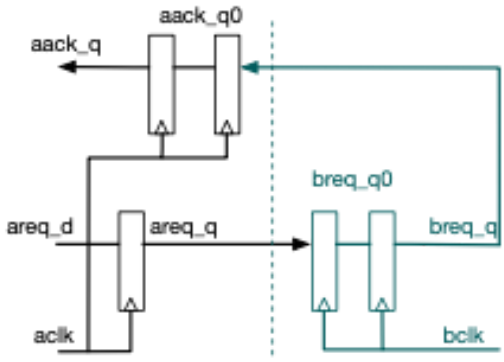
其实使用double flop来同步，有个最基本的“3个沿”要求，就是source data必须保证稳定不变至少碰见destination clock 3个连续的沿，这个沿可以是上升沿也可以是下降沿，持续3个沿之后才能变，否则就有可能在destination clock domain根本看不到这个data的变化。例如下图所示：adata第一次变高，只碰到了bclk的2个沿，就可能导致bdata根本没有看到这个pulse，而第二次adata变高，持续了3个沿，这样bdata就能够确保也可以变高了。

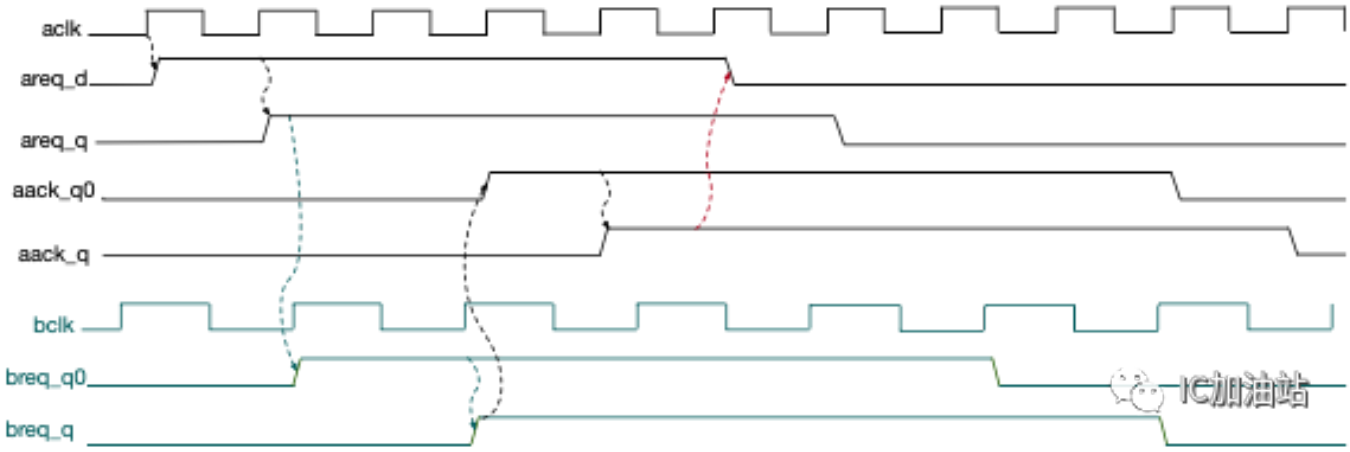


所以如果bclk的频率是1.5倍的aclk频率以上，即使adata是aclk域的一个短pulse，也可以保证3edge要求。如果没有这样频率的关系，那就得对adata有要求了，adata的变化不能很迅速，要稳定足够长的时间，这样才不会让bclk域错过值，具体怎么做呢？在回答这个问题之前，我们应该先停一下，问自己一个问题：我是不是要将adata的每一次翻转都同步到bclk呢？

其实这个问题应该是在考虑要将一个信号跨到另一个时钟域的时候首先要问自己的问题。大多数情况下，回答都是肯定的，也就是说adata变化了，bclk这边的信号也要变化。但是也有些时候，即使漏了adata的变化，可能也没关系，这就是和设计的要求相关了.

如果必须要求adata的每一次翻转都同步到bclk，一个办法就是利用反馈。也就是说信号从aclk域同步到bclk域，再同步回aclk域。aclk的data只有看到同步回来的值之后才能再翻转。如下图所示



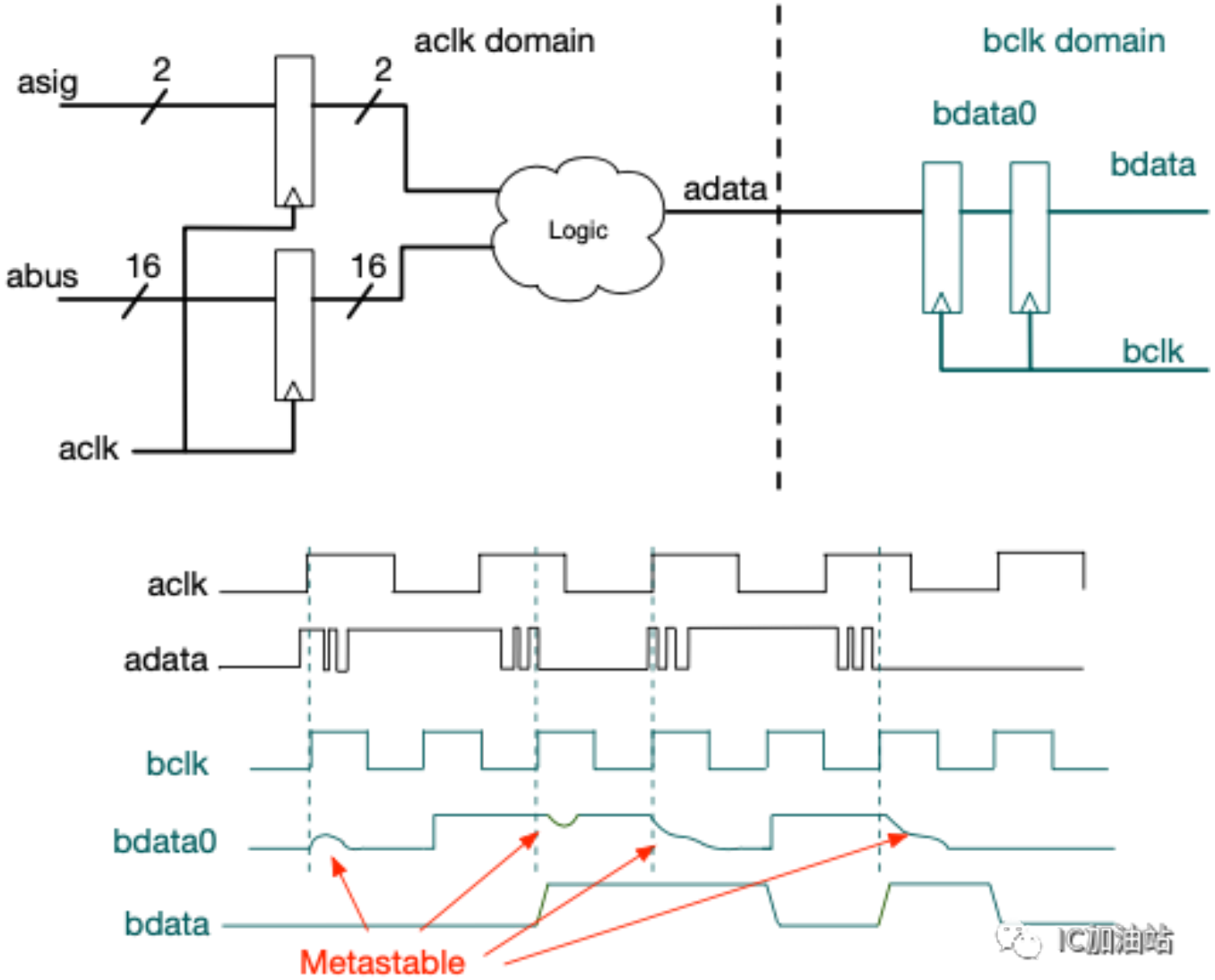


但是这样做的缺点也很明显，就是将aclk的data进行了扩展，两次的同步也增加了延时，这是为了达到每次变化都同步而付出的代价。大多数时候，设计者知道adata变化的频率很低，比如是一个软件配置位，配置好之后可能不轻易更改；比如是一个中断信号，中断发生之后可能需要很长时间才会被软件清除，这些时候就没有必要设计反馈电路了。

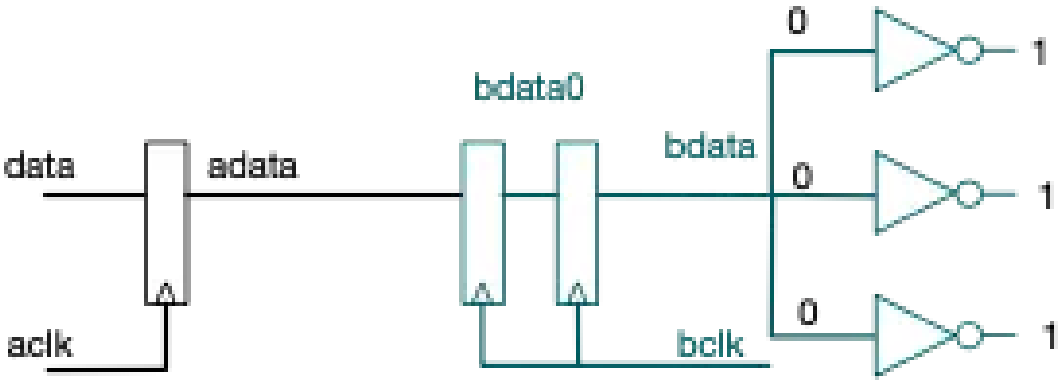
最后再讲两个知识点，也是非常关键的知识点。

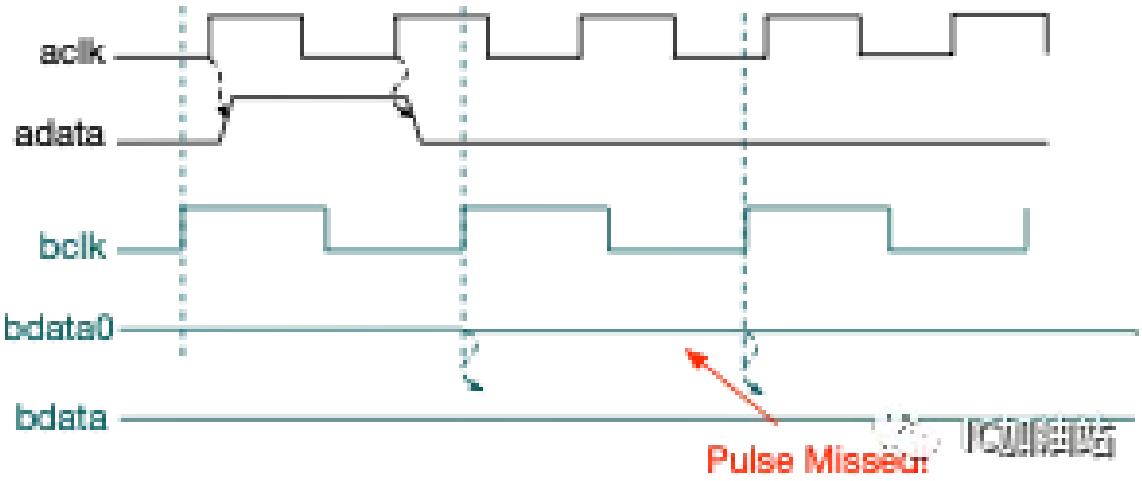
第一，利用double flop，bdata发生变化可能是在adata翻转之后1个周期，也可能是2个周期，这是由于第一级flop的metastable可能会resolve在不同的值。如果第一级flop 稳定在和adata相同的值，那么就只需要1个周期就能看到bdata翻转。而如果第一级flop 稳定在和adata相反的值，那么则需要再多一个周期。所以在设计和仿真验证中，不能假定bdata一定会在2个周期之后发生变化，而是将这个因素随机在仿真中，有的时候真的会暴露出设计中的问题。

第二，我们说的单bit信号，有人可能会说，组合逻辑的输出可不可以用double flop呢？比如一个AND门的输出，不也是单bit吗？答案很简单，不可以。原因就是组合逻辑的输出可能会有毛刺，这些毛刺会增大第一级flop产生metastable的概率，进而影响整个synchronizer的MTBF，更严重的问题是由于第一级flop可能稳定在和输入adata不同的值，会导致bdata出现一个不该出现的值（此处感谢叶浩指出）。**所以对于任何单bit信号，在跨时钟域之前一定要先寄存（flop），只有flop的输出才能经过synchronizer.** 下面的图就是不flop的情形。



结尾之前，要重新再讨论一下当初让小李搞懵的那个例子。





其实在这个例子中，adata在aclk域一个周期就翻转了，其实是aclk域的一个pulse信号。对于pulse信号，我们其实不应该用double flop来同步，原因很简单，就是上面所示的可能丢失pulse。那么面对pulse，我们该怎么同步呢，那就且听下回分解吧。

如果你觉得这篇文章对你有所帮助，不妨点个右下角的“在看”，如果能够分享到群里或者朋友圈，老李就更有动力更新了。