多bit信号跨时钟域怎么办？ -- CDC的那些事（4）

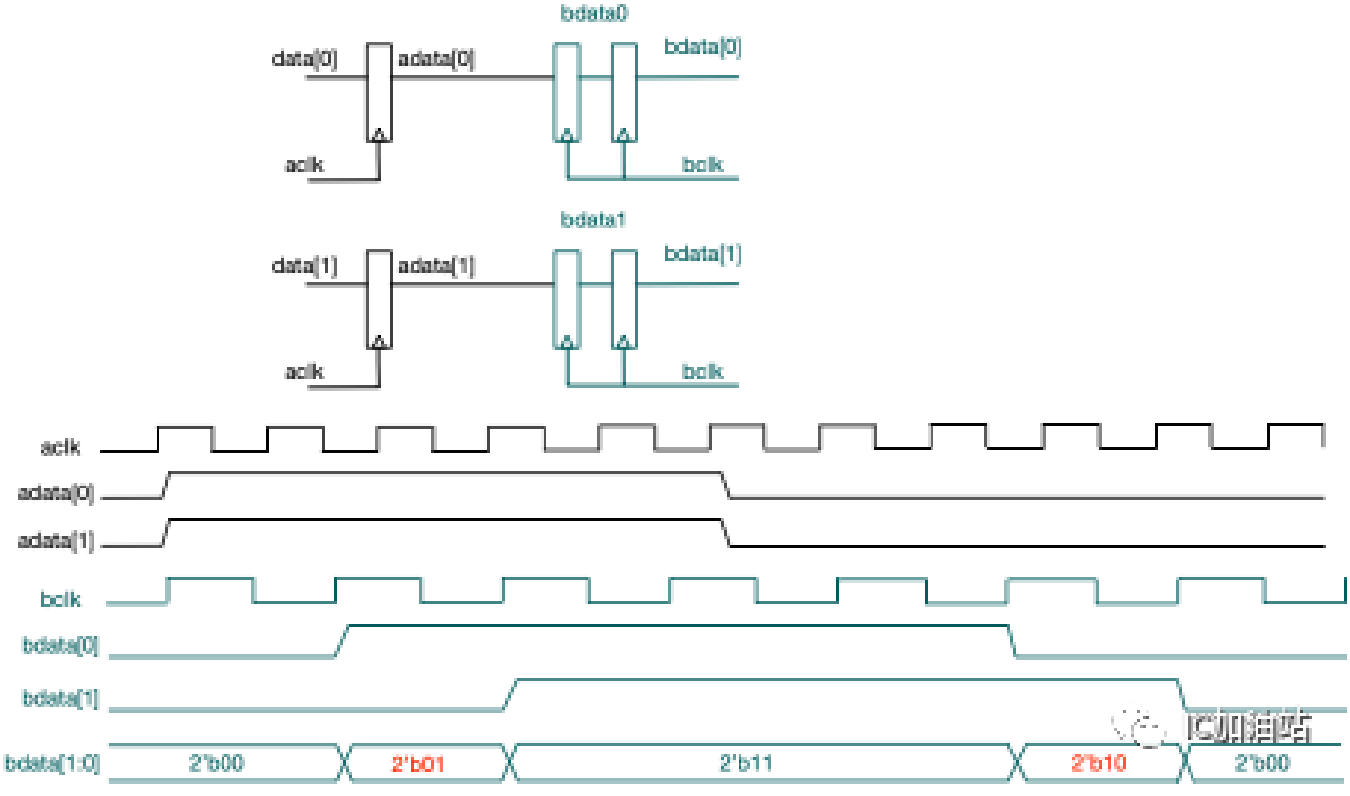
原创 老李飞刀 IC加油站 6月13日

相信经过前面三篇CDC的那些事，大家对于单bit信号的跨时钟域有了相应了解（如果你还没有看，就先看看下面的链接）。下面老李带大家破解多bit信号的CDC。
跟老李一起学习芯片设计-- CDC的那些事（1）
你真的懂2-flop synchronizer吗-- CDC的那些事（2）
常见数电面试题Pulse Synchronizer -- CDC的那些事（3）

这次咱先不废话，直接上一个结论：在绝大多数情况下，我们不能直接利用2flop synchronizer来同步一个多bit信号。（为什么说绝大多数情况，在这篇文章最后我们会讲用2flop synchronizer来同步的例子）。

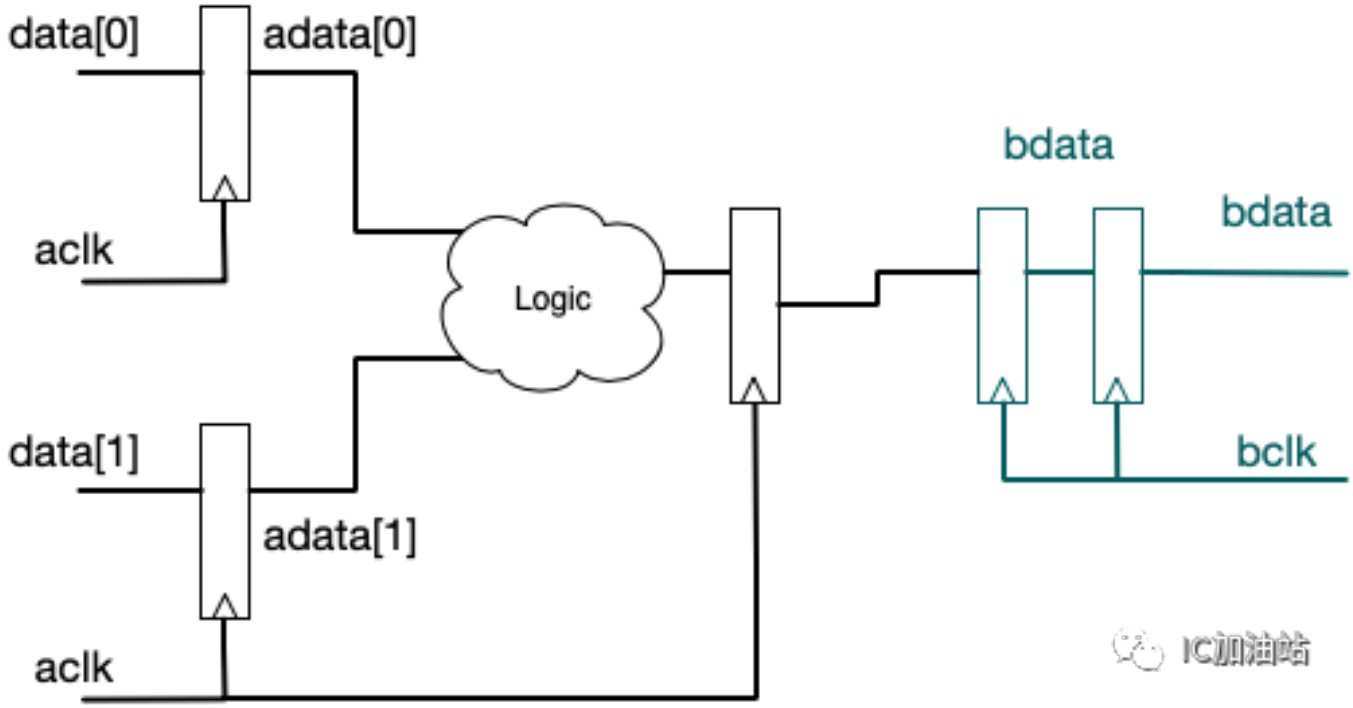
注意我们说多bit信号，是说这个信号是由多于1个bit来表示的。比如说一个couter的值，或者是一个address，再或者本身就是一个多bit的data bus，这些bits之间是相关的，单独拿出来是没有意义的。（有的时候有人会把几个单bit的控制信号group成一个多bit的信号，但实际上各自bit是各自独立的，在同步到bclk时钟域之后也是各自起作用，这种情况其实是“伪”多bit信号，是可以对每个信号用2flop synchronizer同步的，多说一句，这个在CDC检查工具里面大家可以加unrelated attribute来告诉工具这些bit虽然看起来是属于一个多bit信号，但其实是不相关的。）

那么来看一个为什么不能用2flop synchronizer来同步各个bit的例子。



注意adata从2'b00变到2‘b11，一段时间之后再变为2'b00，但是因为2flop synchronizer的delay有随机性，可能是一个周期之后就同步过去了，也可能需要两个周期。这样我们就可能在bdata上看到一个周期的2'b01，之后也可能看到一个周期的2'b10，这两个值都是adata没有出现过的，也就是说bdata出现了错误的值。

那么怎么解决这个问题呢？老李建议大家这个时候先别着急往下看，而是停下来想一想，你是不是真的需要同步一个多bit的信号。有的时候你把CDC的分界线挪一挪，可能你就不需要同步一个多bit的信号了，而是只需要同步一个单bit信号就行，当然要注意同步单bit信号之前要寄存一下。

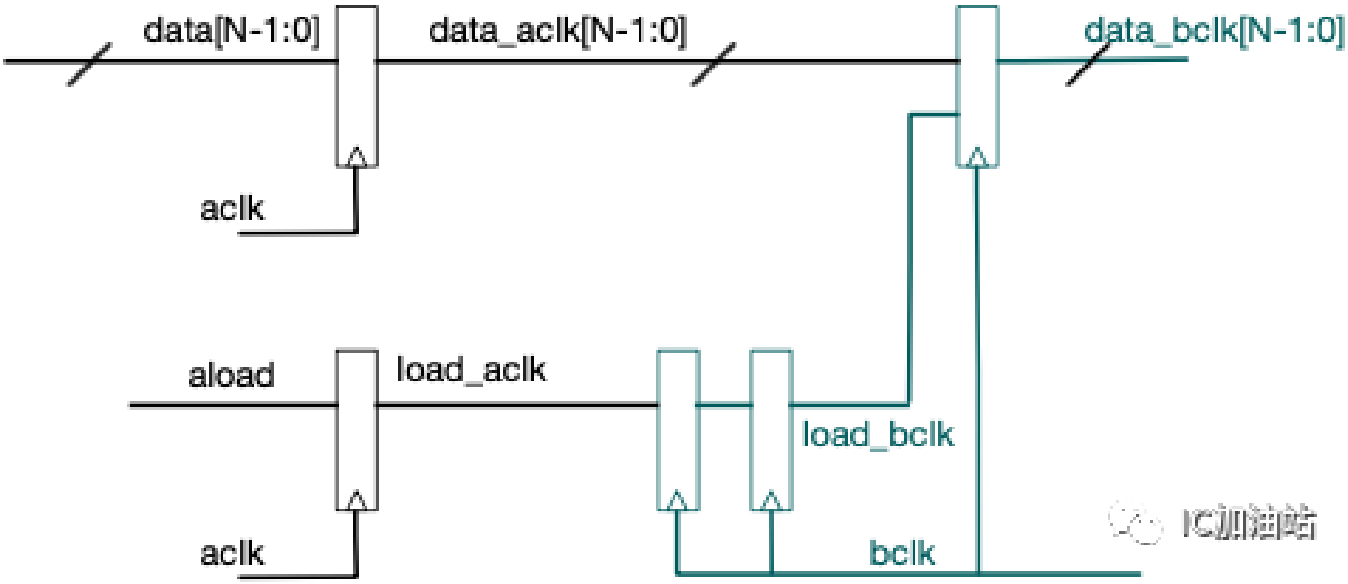


好了，经过你的思考，你告诉老李，不行啊，必须得同步多bit信号过去，咋办呢？

方案一

我们说直接用2flop synchronizer同步多bit信号adata，如果adata的信号在同步的时候变化，就会导致上面出错的问题。那么我能不能想个办法，说bclk在采样adata的时候，adata的所有bit都稳定不变呢？这样就不存在不同bit之间delay cycle不同的问题了。于是思路如下

- 在aclk时钟域产生一个load_aclk信号，load_aclk为1‘b1时代表多bit data信号稳定
- load_aclk信号本身利用double flop同步到bclk时钟域得到load_bclk
- bclk时钟域可以直接利用flop来load bus信号



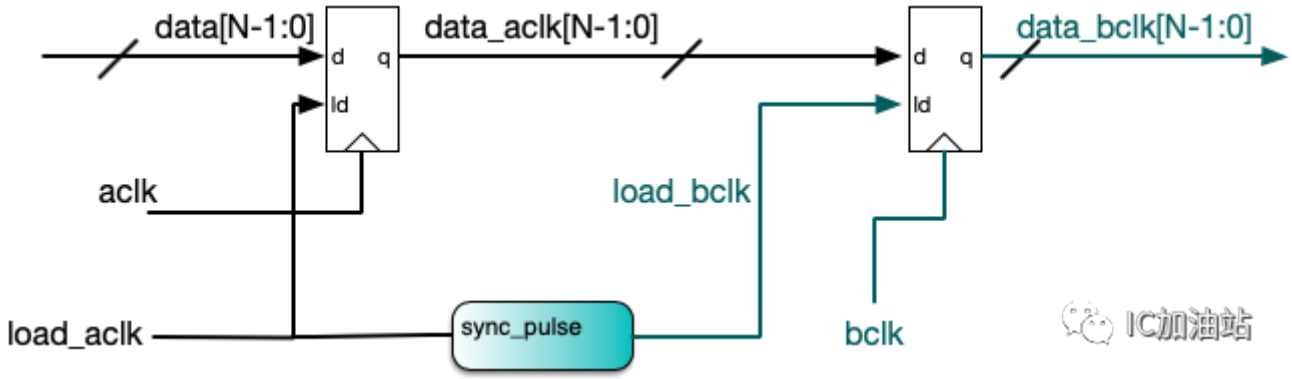
看起来很简单不是吗？但其实这里面有几个隐藏的坑要注意。

第一，要有专门的逻辑保证aload为高的时候data_aclk不变。

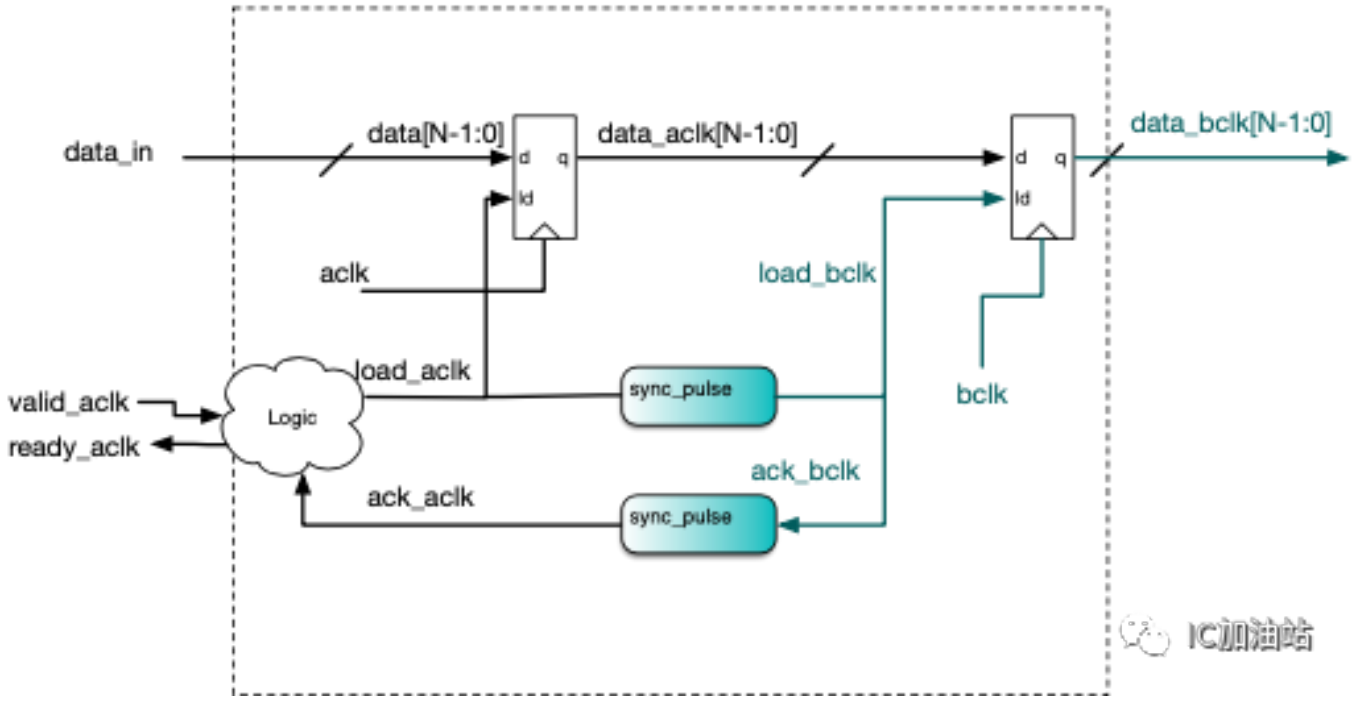
第二，在aload为1'b1的时候，data_bclk会持续load data_aclk, aload从0—>1是ok的，但是1—>0会发生错误，因为data_aclk是不稳定的！

第三，aclk时钟域怎么知道data_aclk已经被成功传到bclk时钟域，从而可以更新下一组data了呢？

我们首先看如何解决第二个问题。我们其实需要的是：当load_aclk变高的时候，把data_aclk当前的值同步过去之后就行了，并不需要持续load。这个时候我们上一篇讲的pulse synchronizer就派上用场了，我们让load_aclk是一个pulse，然后把这个pulse同步过去，这样data_bclk只会load一次。



可是这个还是没有办法解决第三个问题，要解决它，我们只能继续引入反馈大法：把信号从bclk时钟域反馈回来，告诉aclk时钟域load成功，可以更新下一个数据了。如下图所示，aclk时钟域的load_aclk是由一个valid/ready的握手逻辑产生。我们可以把load_bclk再利用pulse synchronizer同步回去，从而让ready_aclk为1，这样我们就知道data_aclk肯定已经被同步到了bclk时钟域，可以更新下一个data了。



似乎大功告成了是吧？别急，再思考一下上面的电路，如果data_bclk的后级还没来得及用它，而data_aclk却开始更新了下一个数，那data_bclk是不是会被新的数覆盖呢？这个问题要怎么解决老李就不直说了，相信聪明的你很快就可以想出来了。

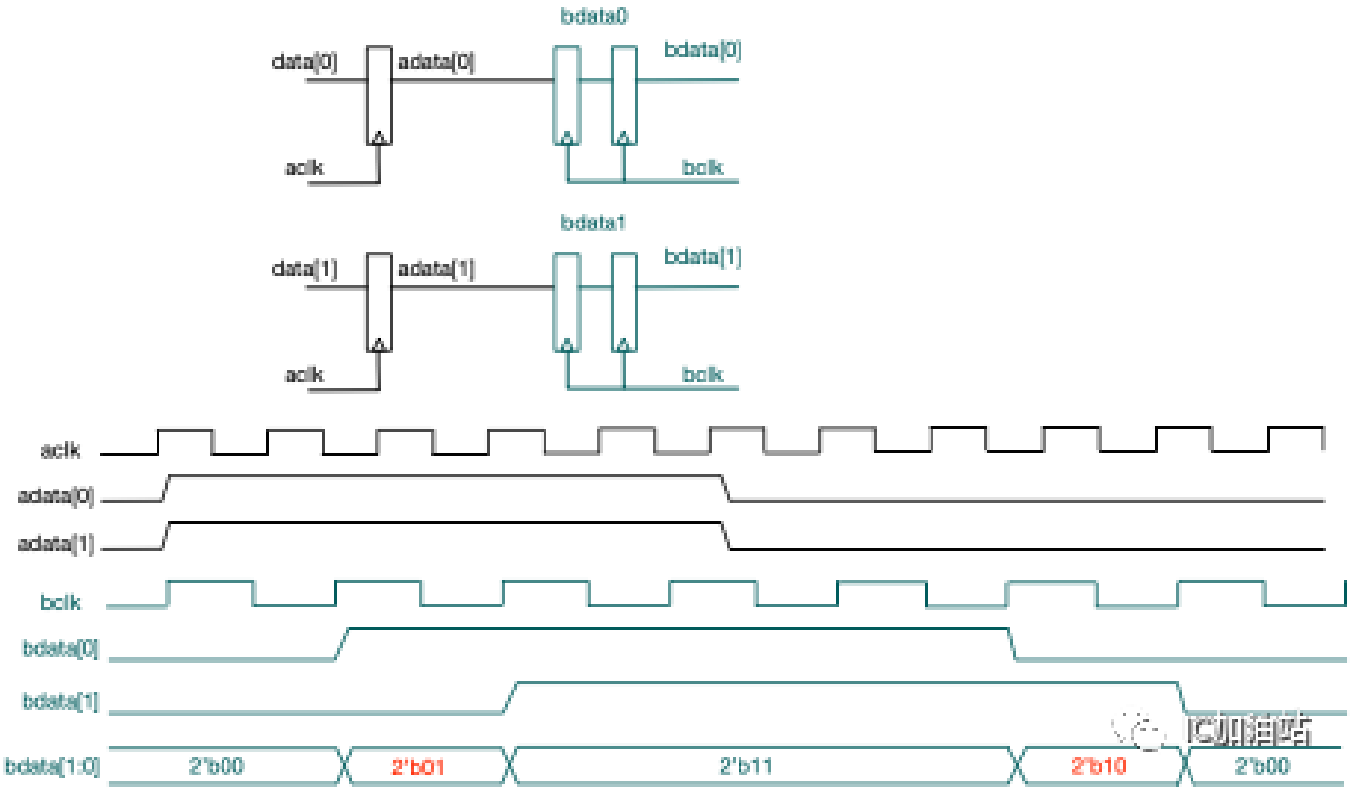
总结一下，方案一适用于

- 无法化简为单bit信号跨时钟域传递
- 适用于非高速传输的场合，即在source 时钟域的多bit信号可以保持稳定一段时间，而不是时刻都在变化，可以有一个明确的load窗口
- load信号为高时必须保证多bit信号稳定不变
- 如果没有连续同步数据的要求，可以适当使用不带反馈的

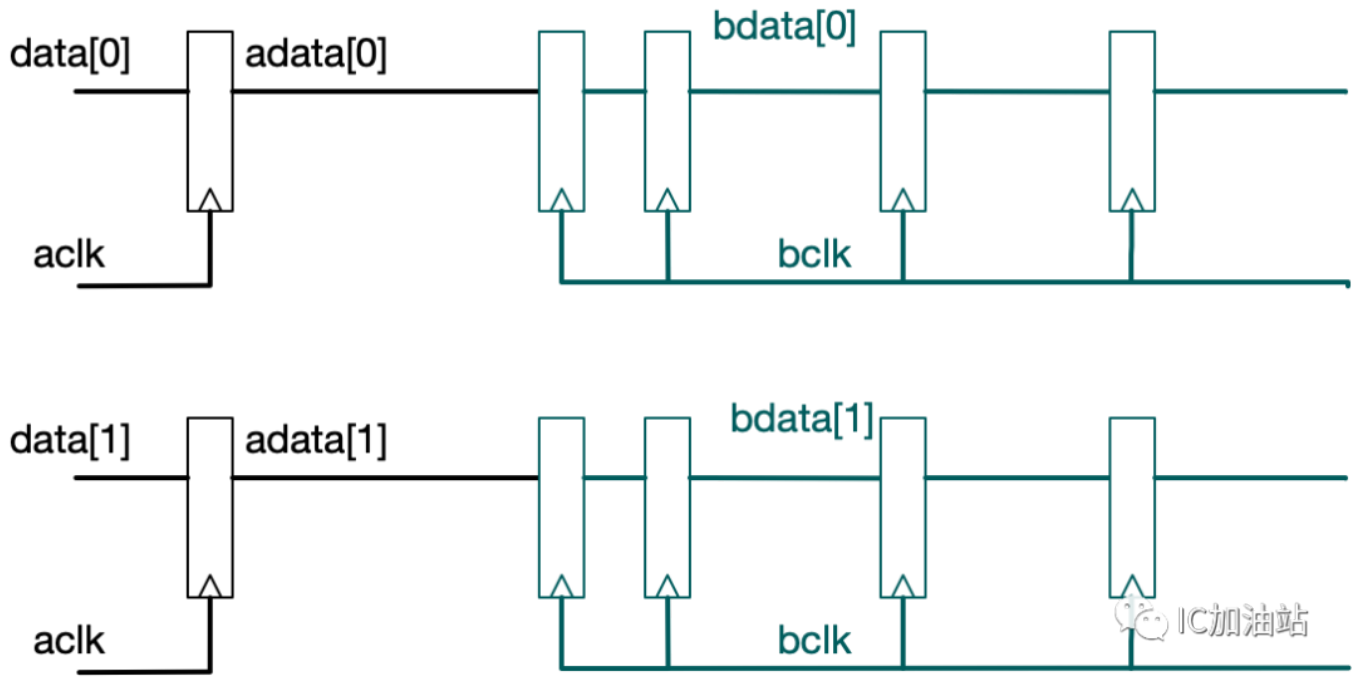
方案二

话说老李当年掌握了方案一，以为掌握了降龙十八掌，加上Asynchronous FIFO就可以横行天下了。结果有一天碰到一个别人设计的模块，要做CDC，多bit信号但是没有load信号，因为多少年的老设计了没人愿意动，这可咋整？

我们再重新看一下上面的那个出问题的图



尽管我们看到了2'b01和2'b10这两个错误的值，但是这两个值中间可是2’b11是正确的值啊，而且2'b11至少持续了3个周期，那么我们其实可以设计一个比较逻辑，利用2flop synchronizer同步到bclk时钟域之后，再用两级flop把bdata打两拍，然后比较这3级的值，如果这三级flop的值是相同的，那不就证明2flop synchronizer同步到的值是稳定的吗？我们可以用三级flop的值相等作为一个update信号，来updata最后输出级的flop（输出级没有画）。



当然我们需要注意的是，这样做的话要求adata变化不是很频繁，因为bclk这边要等好几个周期去比较值是不是稳定，如果adata变化非常快，可以想象，bclk这边的三级flop可能始终没有办法达到彼此相等，从而就无法更新输出级了。

这也是这个方案的缺点，和方案一中没有反馈的结构一样，无法保证每次adata的变化都被bclk实际同步到了。但是如果你确实知道adata变化频率很低，每变一次之后会稳定很长时间，或者说bclk这边不在乎是不是错过了些data，那么你确实可以用方案二。方案二是当你没有办法拿到aclk域的load信号时的back up方案，所以你必须深刻了解它的限制条件。最后还有一点，方案二需要很多级flop，三级flop可能有的时候还不够，要具体分析，但是很明显方案二需要的flop数目更多，尤其是bit数大的时候，面积的花费可能要更高。

方案三就是异步FIFO了。Asynchronous FIFO里面的知识点太多了，值得再来两篇好好探讨一下。下周咱们不见不散。

最后把文章开始埋的坑填一半，什么情况下可以用2flop synchronizer来同步多bit信号呢？其实方案二已经算一种了，还有一种情况就是这个信号是格雷码编码，这个咱们下一篇细细讲。

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”，最好能够分享到群里或者朋友圈，你的支持就是老李更新的动力！