芯片设计进阶之路——跨时钟域深入理解

Rev 1.0 by 烃围玮未 2020/03/06

跨时钟信号处理方法

芯片设计进阶之路——跨时钟域深入理解（2）

在《芯片设计进阶之路——亚稳态和同步器》中，分析了亚稳态和同步器，我们继续理解跨时钟信号处理的各种方法。

跨时钟域的信号可以分为单 bit 信号和多 bit 信号，处理方法有所不同，我们先从单 bit 的 CDC 开始。

CDC: Clock Domain Crossing, 跨时钟域。下面会用 CDC 来指信号跨时钟域处理。

1. 单 bit 信号跨时钟域的处理

信号跨时钟域，根据两个异步时钟之间的关系可以分为：

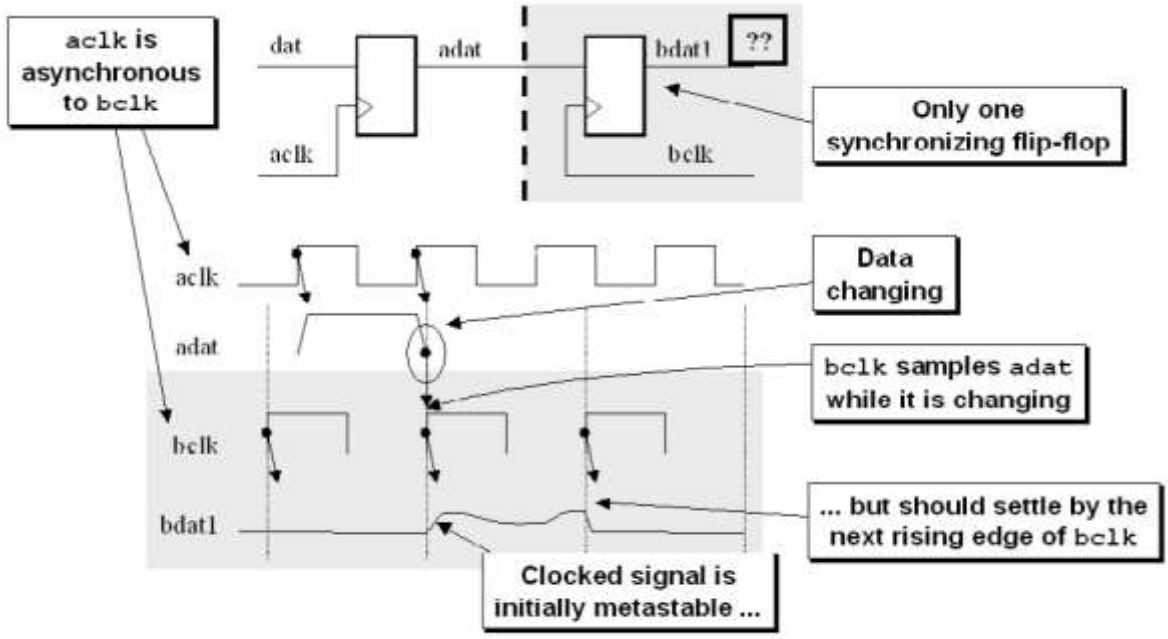
1) 信号从快时钟域到慢时钟域；
2) 信号从慢时钟域到快时钟域；

单 bit 信号一般采用同步器来做 CDC。这里要指出的一点是，由于在 CDC 时，会在源时钟域做寄存输出，所以信号的变化频率不会超过源时钟的频率（请参考《芯片设计进阶之路——亚稳态和同步器》），所以这里能够以两个时钟之间的快慢来分类。

1.1 **快时钟域到慢时钟域**

1.1.1 快时钟到慢时钟 CDC 问题

信号从快时钟到慢时钟 CDC 如下图所示



Asynchronous clocks and synchronization failure

上图显示了当在一个时钟域(aclk)中生成的信号 adat 被送到了另一个时钟域(bclk)中采样，由于采样时间太靠近第二个时钟的上升沿时，发生的同步失败。同步失败是由于输出 bdat1 变为亚稳态，而在 bdat1 再次被采样时没有收敛到合法的稳定状态。

这里注意一下，如果是电平信号进行 CDC, 那么不用考虑时钟快慢，直接用同步器就可以了，因为总能被采样到。所以，下面考虑的主要是信号位宽有限的 CDC。

快时钟到慢时钟的(单 bit)信号处理，主要问题就是信号在快时钟域中，可能会多次改变，这样慢时钟可能来不及采样，导致丢失数据。这个问题被称为信号宽度问题，在 CDC 检查工具中，如果快时钟的信号宽度不足，会报出 CDC 违例。

快时钟到慢时钟的(单 bit)信号处理分为两种：

1. 采样丢失是被允许的。单 bit 信号一般不会是这种情况，如果是这种情况，直接用同步器同步就可以了。

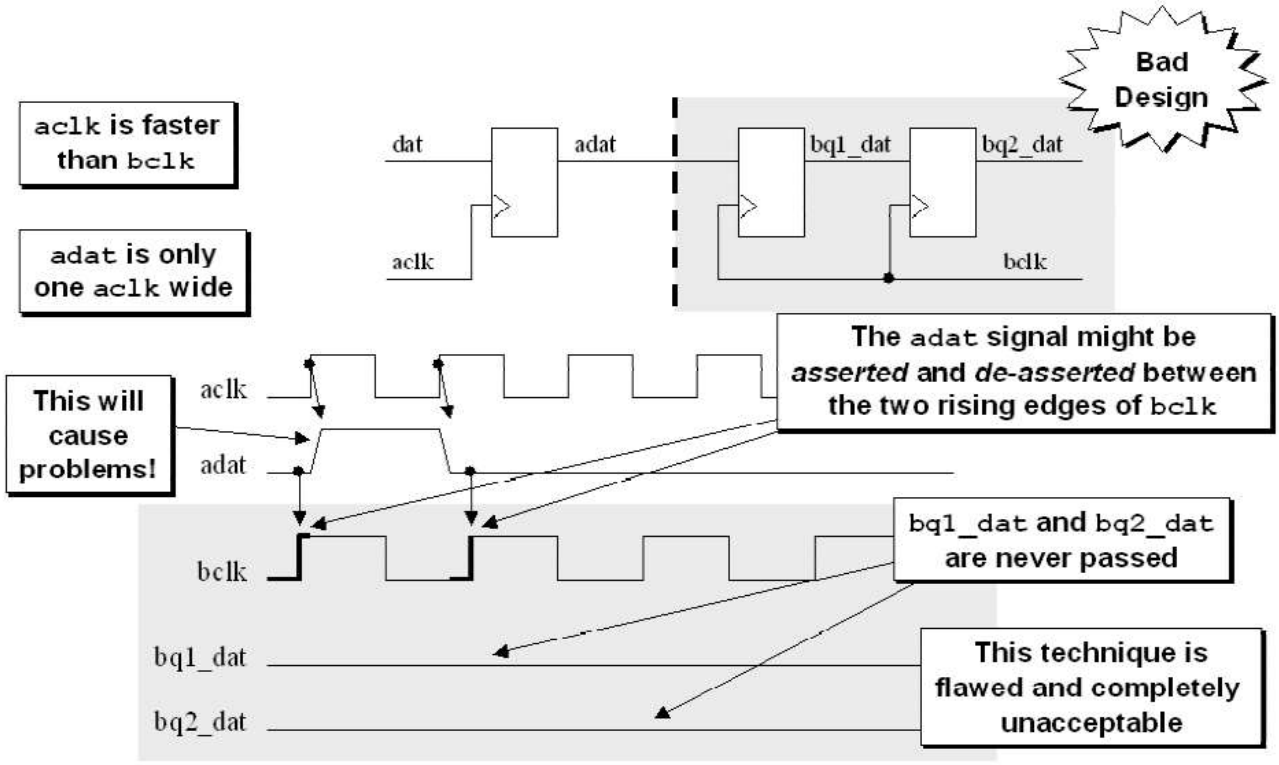
2. 采样丢失不被允许。这样就要采样其他手段来保证数据不丢失。主要原理是保证快时钟域的信号宽度满足一定的条件，使得慢时钟域有足够时间采样到。

1.1.2 信号宽度的“三时钟沿”要求

那么信号在快时钟域到底需要多宽，才能保证在慢时钟域安全的被采样到呢？比较安全的宽度是，快时钟域的信号宽度必须是慢时钟域时钟周期的 1.5 倍以上。也就是要持续 3 个时钟沿以上(上升沿和下降沿都算)。这个被称为：“三时钟沿”要求。

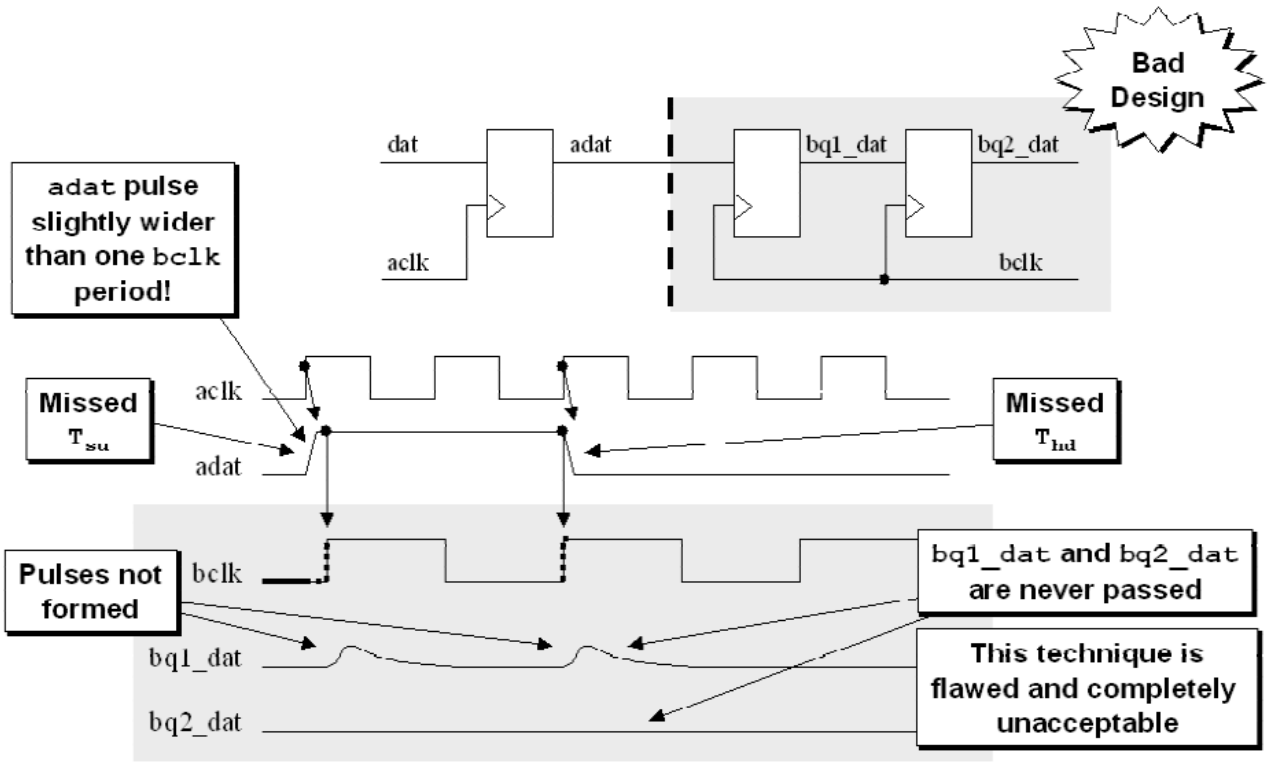
下面是一个 CDC 信号只持续一个周期的例子：

发送时钟域的频率高于接收时钟域，而 CDC 脉冲在发送时钟域中只有一个周期宽，这样 CDC 信号可以在慢时钟上升沿之间变动，不会被捕获到慢时钟域，如下图所示



Short CDC signal pulse missed during synchronization

如果 CDC 信号宽度超过慢时钟周期，但是不足 1.5 个周期，也会发生问题。如下图所示



Marginal CDC pulse that violates the destination setup and hold times

如上图所示，信号宽度超过了一个慢时钟周期，但是可能会 setup 和 hold time 违例，导致信号采样失败。

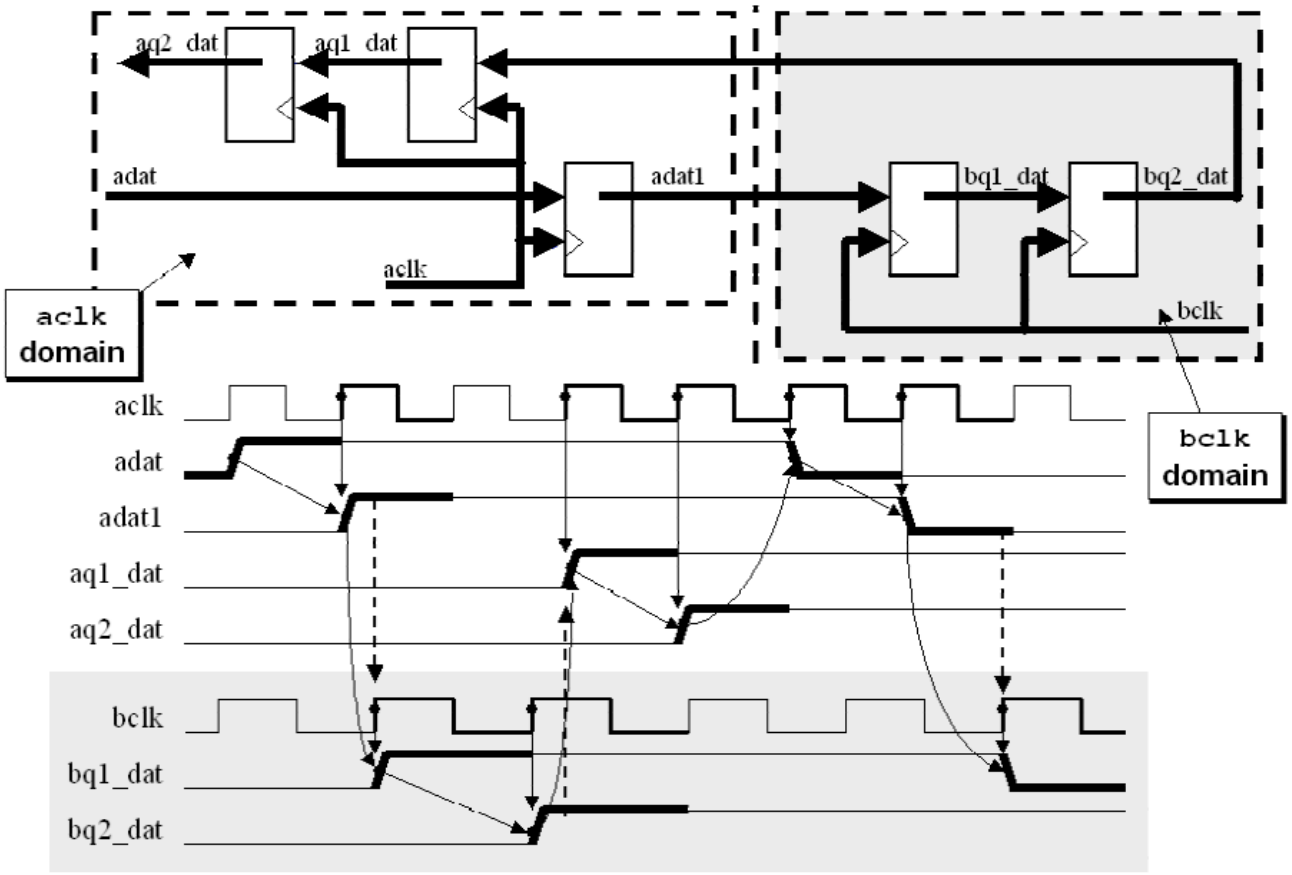
1.1.3 信号宽度问题的解决方法

一种最简单的方法是，通过保证信号宽度满足超过慢时钟的时钟周期 1.5 倍，来解决这个问题。这种方法是最直接，也是跨时钟最快的方法。可以通过 system Verilog 加“断言”的方式来检测是否满足条件。

但是实际中很少用这种方式，因为设计可能会变，设计人员在改变设计时，可能会忘记这个限制，以为是一个通用的解决方法。

所以，常用的还是通过“握手”的方式来保证数据被采样到。

通常的做法是：是发送一个使能控制信号，将它同步到新的时钟域，然后通过另一个同步器将同步信号作为确认信号传回发送时钟域。如下图所示：



Signal with feedback to acknowledge receipt

优点:

同步反馈信号是一种非常安全的技术，可以识别第一个控制信号并将其采样到新的时钟域中。

缺点:

在允许控制信号改变之前，在两个方向上同步控制信号可能会有相当大的延迟。也就是说，在应答信号到来之前，是不允许源信号改变的。

在实际的芯片设计中，脉冲(宽度有限)信号的同步都是采用这种握手机制来处理。

1.2 **慢时钟到快时钟域**

慢时钟域到快时钟域的 CDC， 直接使用信号同步器就可以了。具体逻辑可以参考《芯片设计进阶之路——亚稳态和同步器》。

但是，这里有一点要指出来，那就是怎么才算慢时钟域到快时钟域的 CDC 呢？这里和平常理解的有点不一样。

目标时钟频率必须是源时钟频率 1.5 倍或者以上，才能算慢时钟到快时钟的 CDC.

这也很好理解，只有满足快 1.5 倍以上，才能满足“三时钟沿”的要求，才能保证快时钟域保证能够采样到慢时钟域的脉冲。

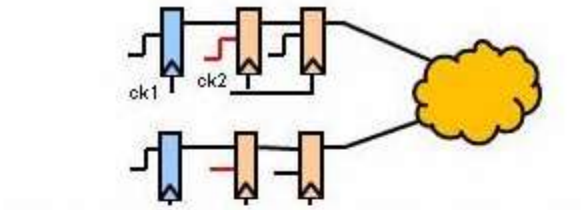
如果目标时钟域只快一点，比如 1~1.5 倍之内，为了保险起见，请按照 1.1 中快时钟到慢时钟域的处理方法来处理。

另外，有的设计中为了保险和以后修改的方便；或者还不清楚时钟之间的关系；都会按照 1.1 中的方式来进行单 bit 的 CDC 处理。

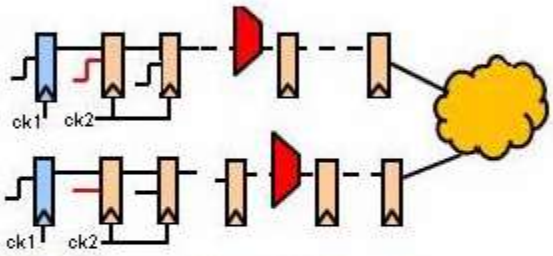
2. 多个信号跨时钟域的处理

在两个时钟域之间传递多个信号，简单的同步器已经不能满足要求。

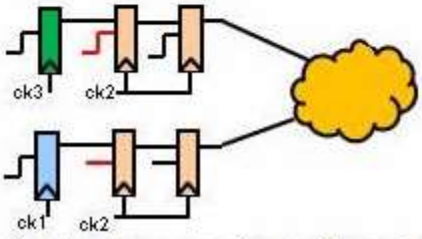
工程师经常容易犯的错误是，直接用简单同步器来同步多个信号。如下图所示：



Combinational convergence from same domain



Sequential convergence from same domain



Combinational convergence from different domains

这里列出理两个信号分别通过同步器同步后，在目标时钟域聚合后使用的三种场景。问题是两个同步器，跨时钟的延时可能不一样，比如信号从 2'b00->2'b11, 上面的同步器花了 2 个周期同步到了目标时钟域；下面的同步器花了 3 个周期才同步到目标时钟域。那么在第二和第三周期之间，就出现了 2'b10 的值了，即出现了错误的采样信号，这样功能有可能就不正确了。

如果不明白，为什么同步器的同步延时会不一样，可以参考可以参考《芯片设计进阶之路——亚稳态和同步器》。

另外，不同同步器的芯片上的走线也可能不同，导致延时不一样。即使我们完美的控制后端不同 bit 同步信号的走线长度一样，同一个 die 上面不同芯片之间或者不同制程之间的偏差，都可能引入差异，导致多 bit 信号的延时不同。而且这样对后端走线难度增大。

考虑到以上两点，**多个信号的 CDC 一般不用简单同步器的方法。**在 CDC 检查时，会有专门的规则来检查是否采样了多 bit 信号用同步器同步聚合使用的情况。

为了避免多位 CDC 倾斜采样的情况，多个信号 CDC 策略可以分为三种:

(1)多个信号合并。在可能的情况下，将多个 CDC 位合并为 1 位 CDC 信号。

(2)多周期路径法。使用同步负载信号安全地传递多个 CDC 位。

(3)使用格雷码传递多个 CDC 位。

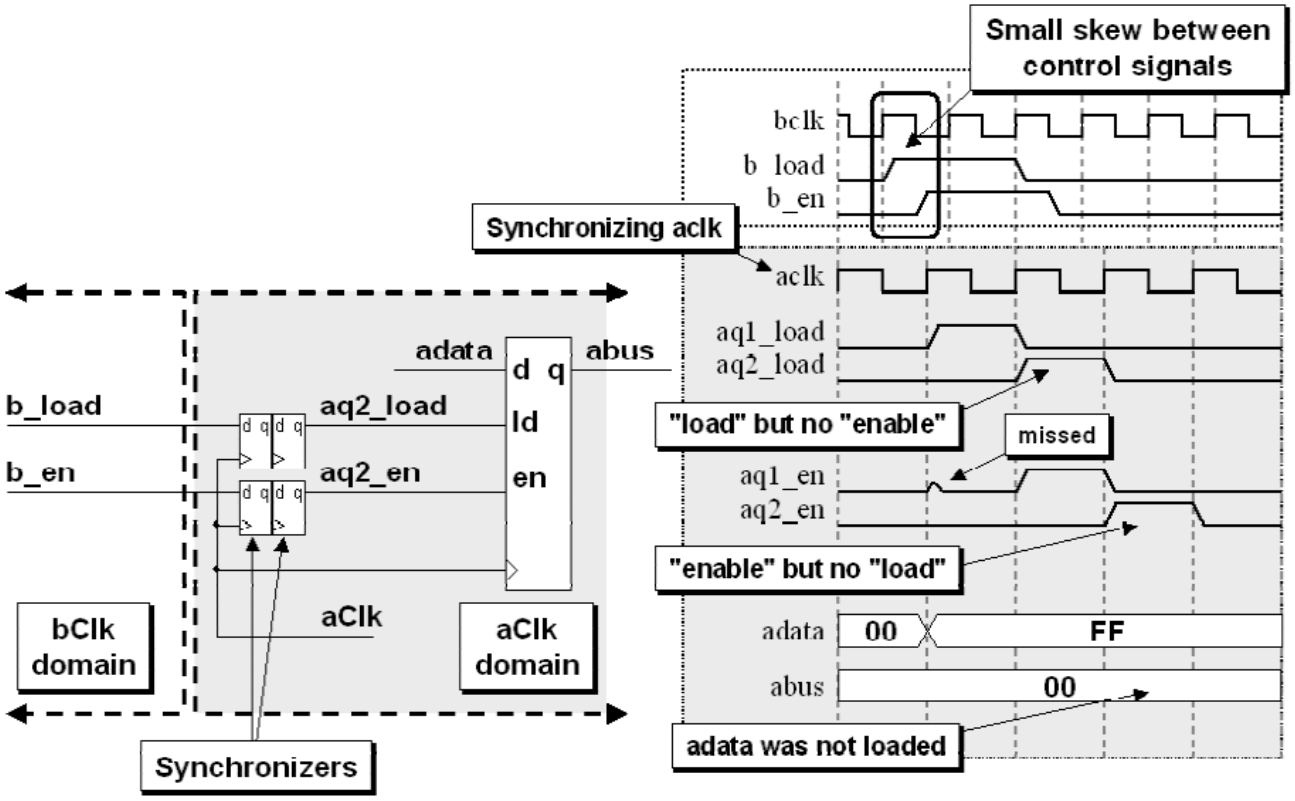
(4) 使用异步 FIFO 来传递多位信号。

2.1 多个信号合并

在可能的情况下，将多个 CDC 信号合并为一个 1 位的 CDC 信号。有些时候，并不需要将多个信号来做 CDC。下面是两个多个信号合并为一位信号做 CDC 的例子。这里

例子 1：

下图所示的简单示例中，接收时钟域中的寄存器需要加载信号和使能信号才能将数据值加载到寄存器中。如果负载和使能信号都是在同一个发送时钟边缘上驱动的，那么控制信号之间的小偏差就有可能导致两个信号在接收时钟域中同步到不同的时钟周期。在这种情况下，数据不会被加载到寄存器中，就会出问题。



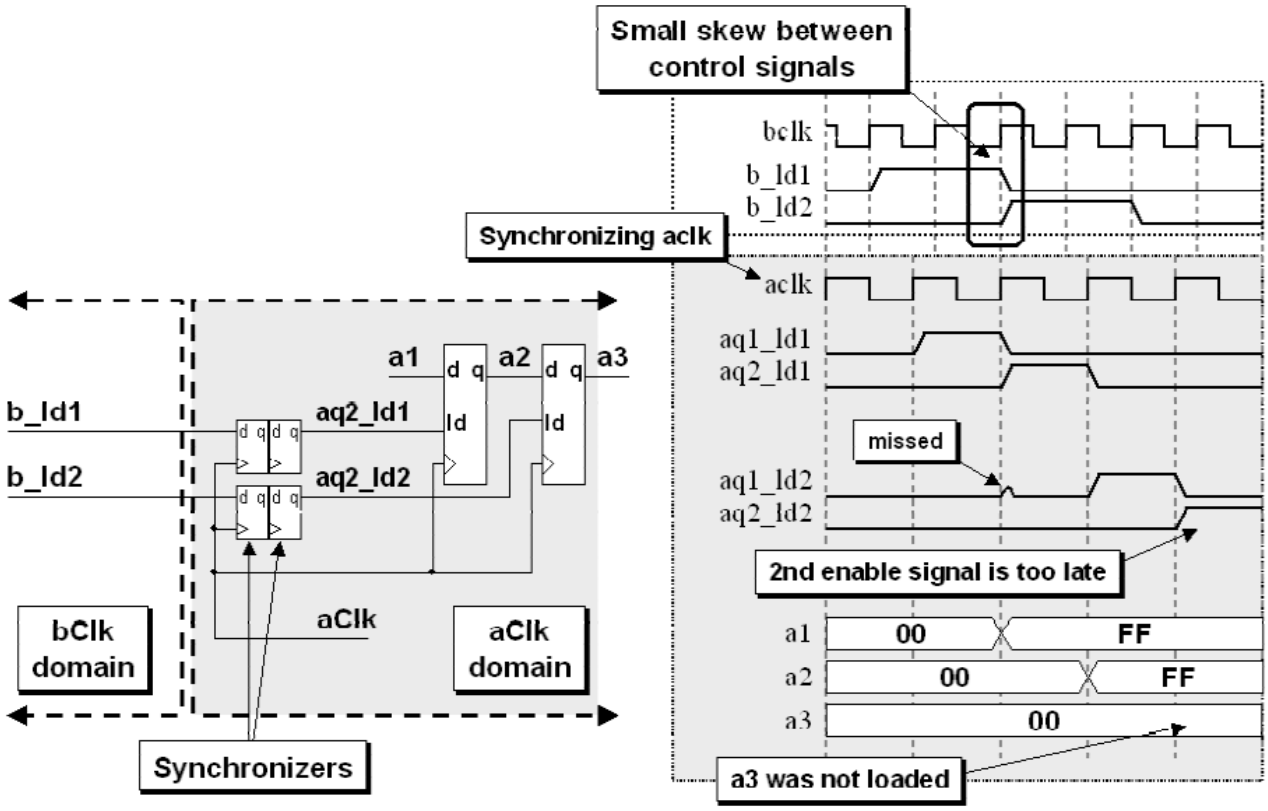
Problem - Passing multiple control signals between clock domains

解决方法也很简单，将控制信号 b_load 和 b_en 合起来：b_lden=b_load & b_en 同步到 aclk 域中。

例子 2:

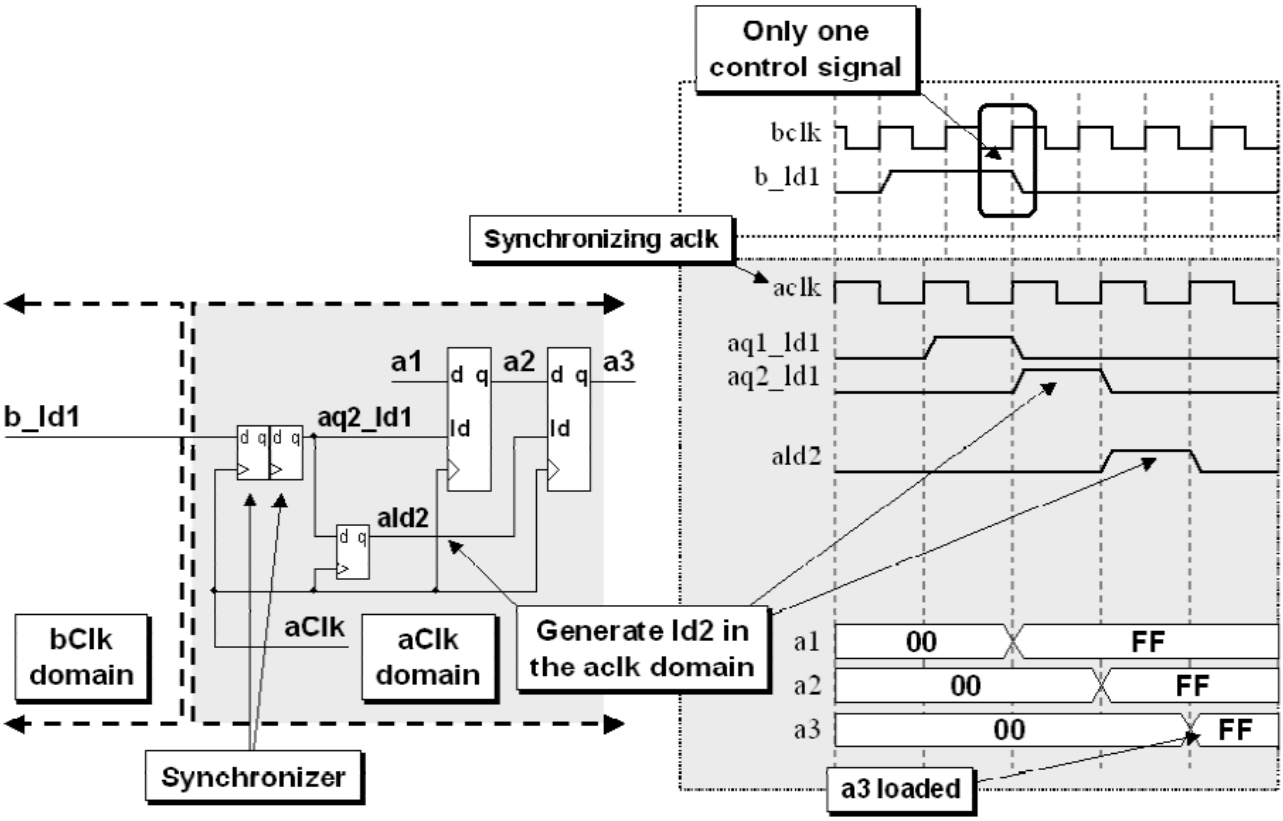
下面是另外一个例子:

下图中的显示了两个使能信号，aen1 和 aen2，它们从发送时钟域依次驱动到接收时钟域，以控制流水线数据寄存器的使能输入。问题是同步器并不能保证两个 cycle 就一定能同步过来，下面的同步器花了 3 个 cycle 才同步完成数据，导致流水线寄存器不能“流水”起来。

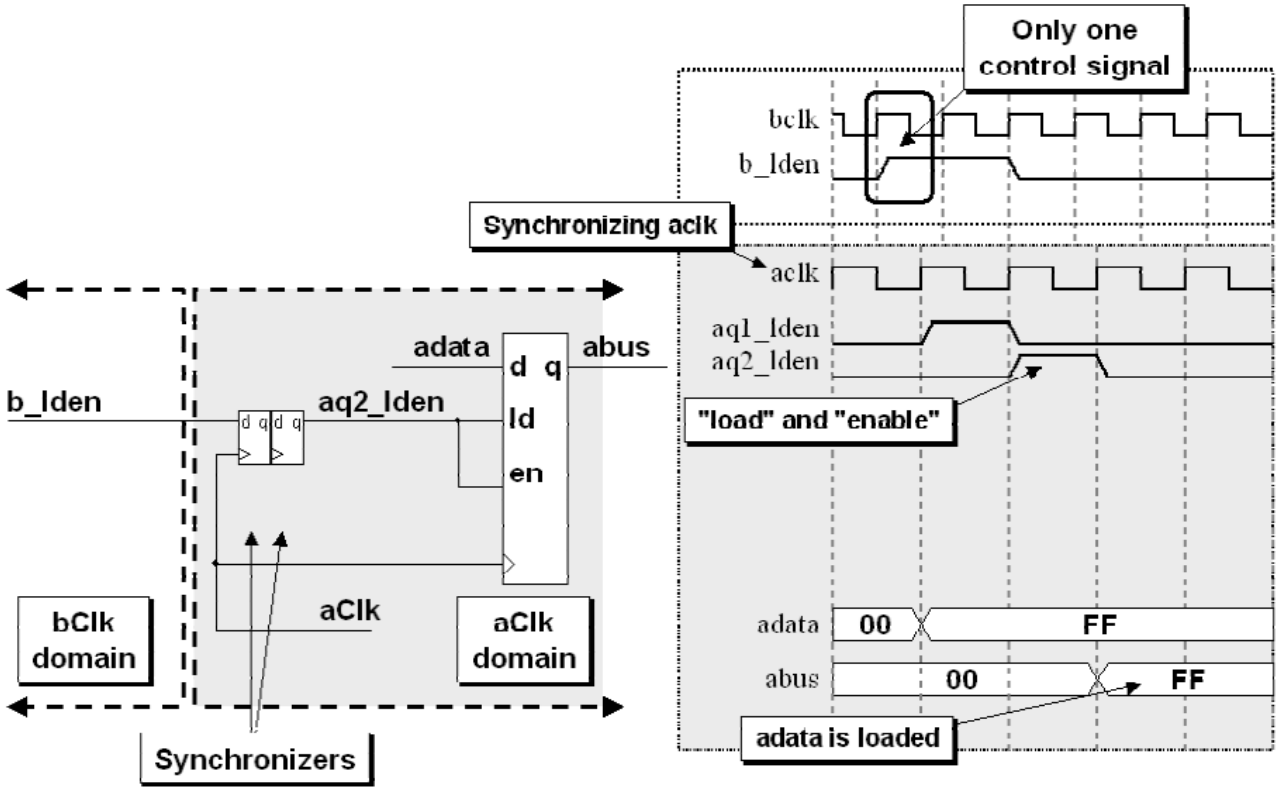


Problem - Passing sequential control signals between clock domains

该问题的解决方案是只向接收时钟域发送一个控制信号，并在接收时钟域内生成第二个相移流水线使能信号。如下图所示：



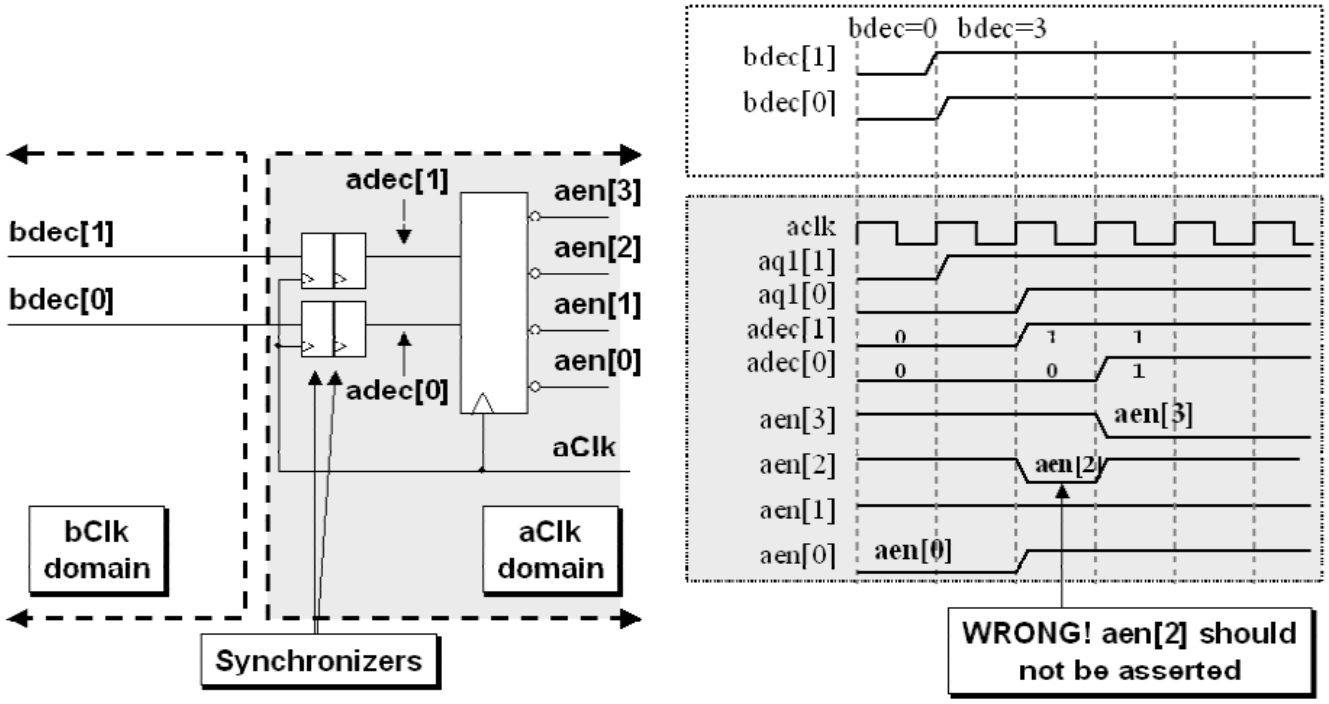
Solution - Logic to generate proper sequencing signals in the new clock domains



Solution - Consolidating control signals before passing between clock domains

2.3 **多周期路径**（Multi-Cycle Path, MCP）

下图中显示了在时钟域之间传递的两个编码控制信号。如果这两个编码信号在采样时略有偏差，则在接收时钟域中的一个时钟周期内可能会产生错误的解码输出。



Problem - Encoded control signals passed between clock domains

多位数据问题可以用“多周期路径法(MCP)”来解决。

MCP 方法是指直接不同步将数据发生到目标时钟域，但是同时送一个同步过的控制信号到目标时钟域。数据和控制信号同时发送，允许数据在目标寄存器的输入端进行设置，同时控制信号在到达目标寄存器的负载输入端之前做同步。

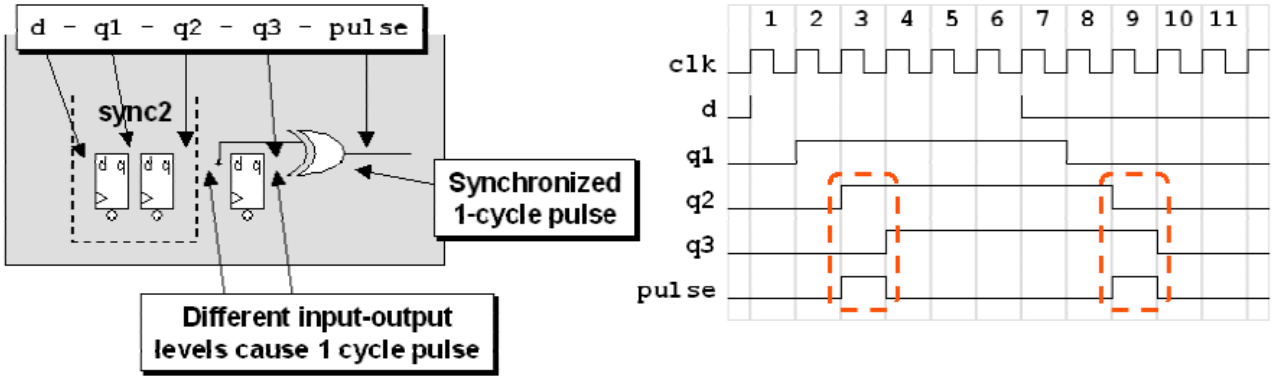
MCP 方法的优点:

(1)不需要在发送时钟域计算适当的脉冲宽度

(2)发送时钟域只需要将 enable toggle 到接收时钟域，表示数据已经被传递完成，已经准备好被加载。使能信号不需要返回到初始逻辑电平。

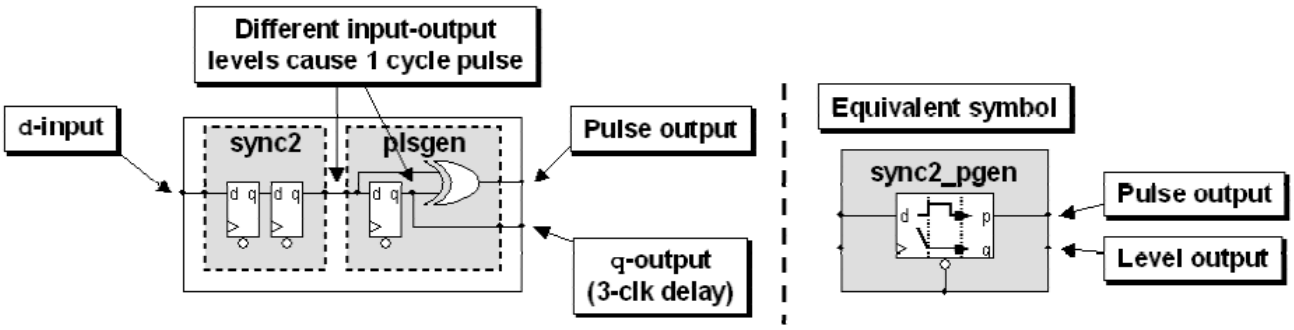
MCP 方法的实质就是，不同步多位的数据，只同步一位的控制信号，通过握手保证控制信号能够正确传输，然后在目标时钟域通过控制信号来采样数据。

MCP 需要用到“同步脉冲器”：



Synchronized pulse generation logic

同步脉冲器的符号表示如下：

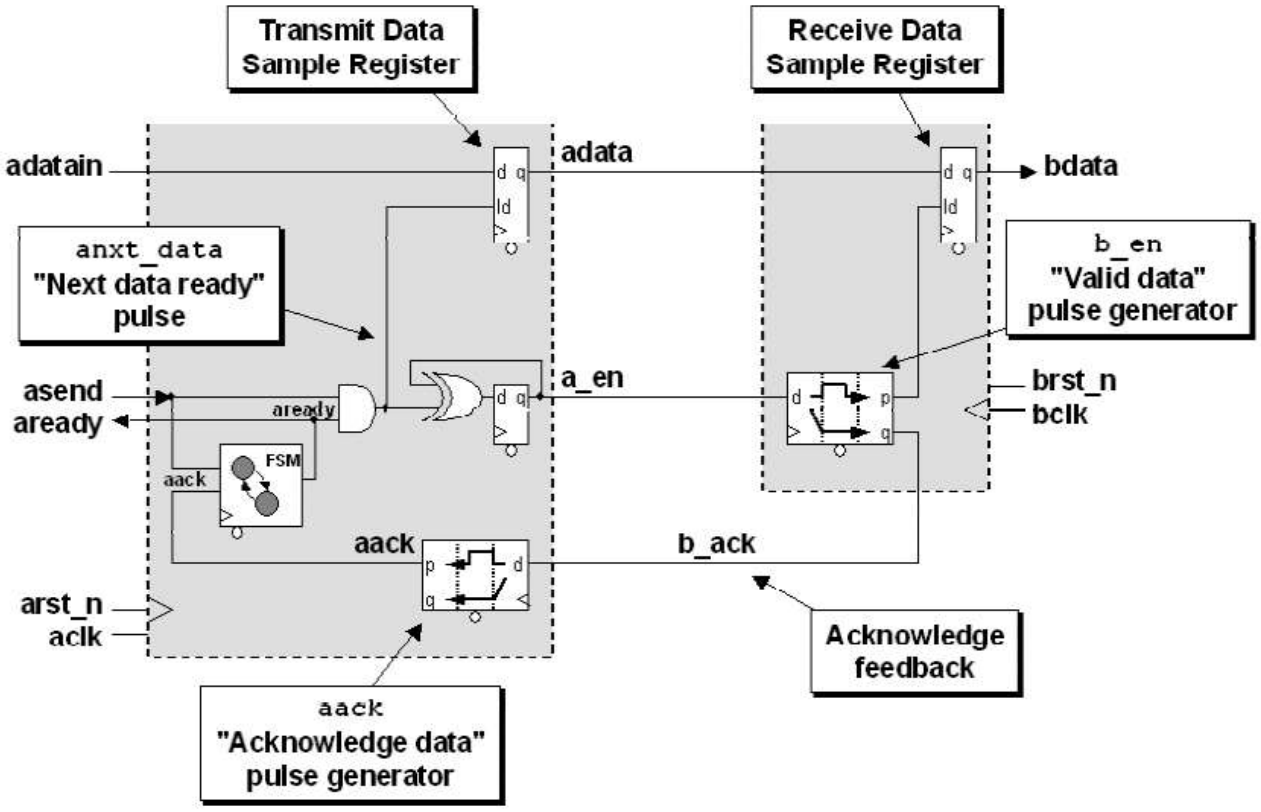


Synchronized enable pulse generation logic and equivalent symbol

多周期路径法有两种方法来传递多位信号：

1. 带反馈的 MCP

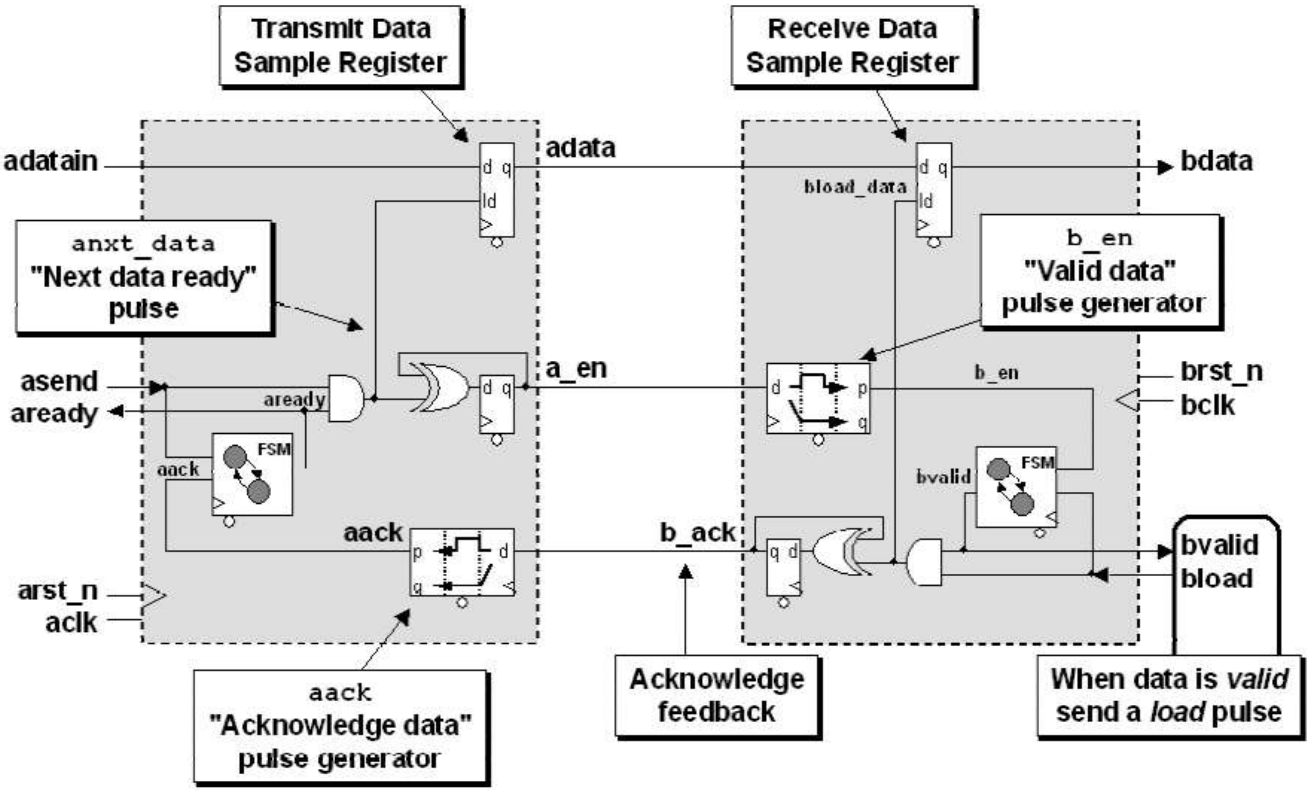
电路图如下



Multi-Cycle Path (MCP) formulation toggle-pulse generation with acknowledge

2. 带应答反馈的 MCP

电路图如下：



Multi-Cycle Path (MCP) formulation toggle-pulse generation with ready-ack

多周期路径法的思想十分有用，但是实际中用来传递多位信号比较少见，因为逻辑过于复杂。但是 MCP 方法用来传递单 bit 的信号却十分有用。这里就不展开讲了，有兴趣的可以参考最后的参考文档中的描述。

2.4 格雷码

对于计数器的 CDC, 大部分是不必要的。如果一定需要，那么可以使用格雷码。

格雷码每次只允许更改一个位，从而消除了跨时钟域同步更改多个 CDC 位所带来的问题。

格雷码和二进制码之间的转换是一种很成熟的技术，很容易就能找到现成的代码，这里就不在详细描述。

需要注意的是：格雷码必须是计数到 2^n 才是每次改变一个 bit。

如果计数器是从 0~5 计数，那么从 5->0 的计数，不止一个 bit 改变，就失去了只改变一个 bit 的初衷。

格雷码最常见的应用是在异步 FIFO 中，通常异步 FIFO 的深度都是 2^N，原因就是上面说的。所以，就算浪费面积，也需要把 FIFO 深度设置为 2^N。

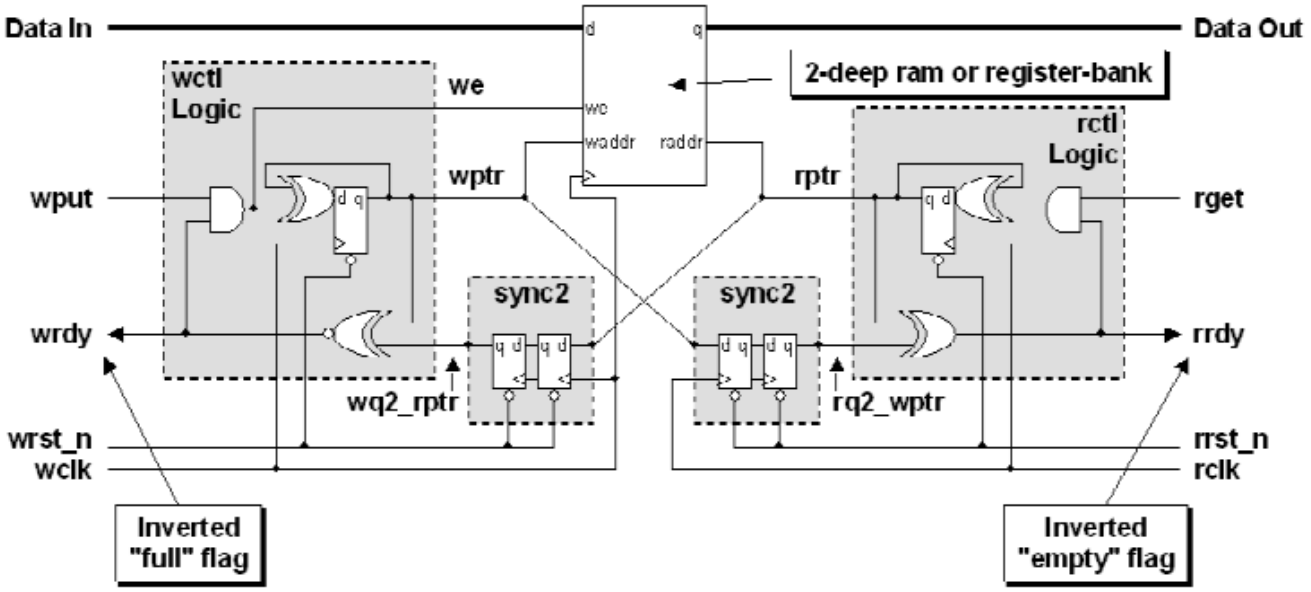
2.5 通过 AFIFO 进行多位信号 CDC

多位信号 CDC 的工程上的一般做法都是采用异步 FIFO, 异步 FIFO 的设计请参考：

“Clifford E. Cummings： Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons”

这篇文章，里面写的十分清楚，而且给出了详细的 RTL 代码，可以直接使用。把这篇文章搞懂了，AFIFO 的知识点基本就清楚了。AFIFO 的设计也是芯片设计的基本功，一定要弄清楚。

这里有一个特殊的应用，那就是深度为 2 的 AFIFO，来进行多 bit 数据的 CDC. 如下图所示：



1-deep / 2-register FIFO synchronizer block diagram

2 个寄存器搭建的 AFIFO，地址只需要一位。相比 MCP 方法，逻辑简单，可以复用 AFIFO 代码（一般公司都有芯片验证过的 AFIFO 代码），而且延时也比 MCP 方法小。

所以多 bit 仅仅跨时钟域，不需要进行数据吞吐率匹配(FIFO 的重要功能之一)的情况，推荐用深度为 2 的 AFIFO 来实现，而不是 MCP 方法。

3. 总结

CDC 问题是芯片失败的最常见的问题之一，弄清楚 CDC 方法也是芯片设计的最重要的事情。常见的 CDC 方法总结如下：

- 推荐的 1-bt CDC 技术：
  1. 慢时钟到快时钟：同步器；
  2. 快时钟到慢时钟：MCP 方法
- 推荐的 multi-bit CDC 技术：
  1. 多个信号合成一个信号。
  2. MCP 方法（其实用的很少，但是设计思想很好）
  3. AFIFO
  4. 2-Depth AFIFO

4. 后记

跨时钟设计是芯片设计的最重要的主题，没有之一。信号的跨时钟处理是芯片设计师必须要要掌握的技术，深入理解 CDC 的问题和对应的方法，是芯片设计进阶的必经之路。本文是跨时钟域（CDC）深入理解文章的第二篇，后续会陆续发布跨时钟域（CDC）深入理解系列的文章，尽情期待。

赠人玫瑰，手有余香。原创不易，如果你有所得，麻烦花一秒时间帮我“**点赞**”吧，谢谢！

知乎专栏：芯片设计进阶之路

微信公众号：芯片设计进阶之路 x_chip

——

参考资料：

Clifford E. Cummings，"Clock Domain Crossing (CDC) Design & VeTechniques Using SystemVerilog”

——