西安电子科技大学

学位论文独创性声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切法律责任。

本人签名：_____ 日 期：_____

西安电子科技大学

关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律署名单位为西安电子科技大学。

（保密的论文在解密后遵守此规定）

本学位论文属于保密，在 年解密后适用本授权书。

本人签名：_____ 导师签名：_____

日 期：_____ 日 期：_____

摘 要

随着数字系统复杂度的提高，系统芯片中集成的模块数量增加。各模块通常工作在不同的时钟频率下，对各系统之间的数据 CDC（Clock Domain Crossing）通信需要进行同步设计。对于不同时钟域和电压域的情况，对 CDC 同步设计的要求不同。对于这些 CDC 传输路径以及同步设计的检查验证，在整个设计流程中的作用日渐凸显。目前还没有一套比较成熟、完善的验证手段，能在设计早期 RTL 级就能完成 CDC 的验证工作。本论文通过分析 CDC 传输中的亚稳态机理，总结了各种同步设计的优劣以及传统验证方法在 CDC 检查中的弊端，提出并搭建了一套基于 SpyGlass 的 CDC 静态验证流程。在 RTL 级脱离验证平台和测试向量，穷举设计中所有 CDC 传输路径并进行了验证。基于不同的验证结果，对设计缺陷提出了建议性修改方案。经过实际项目的检验，证实该验证手段在设计流程中是可行有效的，对 CDC 设计与验证有一定借鉴意义。

关键字：数字集成电路　亚稳态　平均失效时间　同步设计　静态验证

Abstract

So many module blocks are integrated in current digital SOC with the increasing of the complexity. Such modules are all working under different clock frequencies commonly. Clock domain crossing of the data among modules always need to be synchronized by dedicated designs. There are different requirement in the sync designs, with regard to the different clock-domain or power-domain scheduling. So the verifications for such clock domain crossing paths and inserted data sync cells become more and more important in IC design flow. Currently there is stilling not perfect method to solving this issue in early designing stage. All diff sync designs' merits and the traditional verification methods' weakness upon CDC aspect in current flow are made a conclusion based on analyzing of the CDC meta-stability mechanism in this thesis. Meanwhile, a static verification environment and flow based on SpyGlass tool without test bench and test case in RTL level are introduced and built. Based on this new verification method all the CDC paths in design are exhausted and carried on examinations. Some changing suggestions for the common RTL design defects are given as well in the article. It is shown that this static verification method is feasible by application in IC project, which are reference meaning to the CDC design and verification part.

Keyword: **Digital-IC Meta-stability MTBF（Mean Time Between Failures） Synchronization design Static verification**

目录

第一章 绪 论

1.1 CDC 同步与验证研究背景

随着当今集成电路技术的飞速发展，日常生活中充斥着越来越多的微电子商品，从仪表仪器，到电信网络，从工业设备，到日用数码，这种新型技术已经深入人们的生活中。对于各种实际需求的不同和增加，这些集成电路设计的本身规模也在逐渐扩大，功能也逐步繁多。

而集成电路本身是由各种不同的功能电路来构成，电路内部触发器之类时序元件随着独立、一致的时钟节拍来翻转，从而完成数据的采样和传输。这样的时序一致性可以允许集成电路规模的发展扩大,同时这种方式也给设计后端的综合、布局布线带来了很大的方便。然而芯片的功能的不断增强，集成电路必然趋向使用多时钟信号的设计方向，这些不同的时钟信号在频率和相位方面是有很大区别的（即不同源），如果单纯地去使用两个无任何相位关系的时钟信号来采样和传输数据，即数据的跨时钟域—CDC，则会出现一系列时序问题，如触发器亚稳态的产生以及传播，从而引起部分相关电路功能的错误运转，甚至导致整个芯片系统的崩溃。

基于这种多时钟域的设计，通常需要对不同时钟信号间的数据传输进行同步设计，通过同步电路技术，系统中的这些异步电路之间的数据传输所潜在的亚稳态等时序问题将会得到很好的隔离与解决，使数据在整个芯片上不同时钟区域之间的传输更加安全，以此来保证芯片系统的稳定工作。

因此从解决问题的角度出发，就希望能够在 SOC（System On Chip）设计的更早期发现这种潜在的时序风险。从目前大多数传统的动态功能性仿真工具的使用效果来看，对于这类跨时钟域亚稳态问题不能进行充分的验证；而在设计后端的 STA（Static Timing Analysis：静态时序分析）中，对这样的时序问题也不能完全的覆盖检查，一方面因为 STA 只能够检查在同步的情况下电路时序的收敛性，对于 CDC 的异步时序却是存在验证方面的缺陷，另一方面由于目前 SOC 设计的周期在不断的缩短，设计的时间节点和质量都是保证一款芯片占领市场的首要条件，对于 STA 以及后仿等验证，所做的时序收敛性检查往往是在设计流程的后端，然而在此时间节点一旦发现一些的 CDC 时序问题的话，对整个芯片流片以及产品上市都是致命的。

基于目前 CDC 研究背景，发现和定位这些 CDC 问题的工作，毫无疑问的要往设计的更前端移动，从后仿验证者逐步转移至 RTL（Register Transit Level）的

设计人员手中。

1.2 CDC 同步与验证研究现状

同步设计的发展起源于上世纪 60 年代，因其设计相对比较简单，且具有丰富的工程化设计手段而逐渐占领电路设计的主流。上世纪 80 年代开始，跨时钟域信号的传输开始受到国外研究人员的关注，在许多的工程实践中，总结归纳了一些跨时钟域信号传输的同步机制。随着国内微电子技术的发展，跨时钟域设计逐渐才受到国内研究人员的关注，单仍然缺少对此类同步设计的系统研究与总结。

而对于此类 CDC 同步设计的验证工作，虽然各公司设计者们都已经注意到在设计流程的更早期实施验证的方法必要性，但是由于目前国内相对于这种新型的专门验证工具的引进存在障碍，同时对于此类验证方法的研究不能得到实际项目的支持，导致目前国内在该领域仍然存在着许多设计和验证方面的不足。致使大多数项目设计人员只能在设计流程的后端乃至流片之后才能察觉这些严重的 CDC 问题。

对于集成电路设计者来说，能够正确的应用不同类型同步设计，并且有一种专门的 CDC 验证方法才会是最完备的。国际上各大 EDA 公司都在不断地提出和完善自己的 CDC 设计与验证工具，例如 Mentor 公司的 0-In、Synopsis 公司的 Leda 以及 Atrenta 公司开发的 SpyGlass 等工具，而这些 CDC 验证工具都有各自的优势和局限性，在 CDC 验证上的表现也不尽相同。

目前虽然国际上推出了一些此类的验证工具与验证方案，但是由于目前国内在 SOC 设计领域许多客观条件的限制，使得不能够充分的了解与认识这种验证工具的特性，也就不能在实际的项目中建立实施。因此基于此类新型 CDC 工具的大规模数字 SOC 的 CDC 验证环境以及流程仍然十分不成熟。

1.3 论文内容安排

本文在基于上述研究背景和现状，扼要阐述了设计中常见的跨时钟域问题，以及亚稳态 MTBF 随时钟频率和不同的工作电压的变化分析。根据这些分析结果，系统的介绍和总结了在如今系统 SOC 设计时候经常使用的几种同步电路设计的优缺点和应用时候的限制条件，包括在设计中经常被忽略的异步复位-同步释放的设计问题和传统验证方法在 CDC 检查方面的缺陷。同时针对目前国际上的几种 CDC 验证工具，介绍了一种基于 Atrenta 公司的 SpyGlass-CDC 验证方法，阐述了该验证方法以及基本验证规则。最后，基于这种 SpyGlass-CDC 工具，引入设计了一种 SOC 基带芯片的 CDC 验证环境与验证流程，以及针对常见的 RTL 设计缺

陷，给出了一些建议性的修正方案。具体内容如下：

第一章，绪论。

第二章，概述了数字 SOC 设计中常见的 CDC 问题，以及亚稳态原理和触发器亚稳态 MTBF 数学模型，根据目前 SOC 设计中常见的 clock-domain（时钟域）以及 power-domain（电压域）划分情况，着重分析了亚稳态 MTBF 在此类设计架构下面的变化情况；并且简述了这些问题在设计中的危害，以及这些问题对整个设计流程的影响。提出一些高速、低功耗 SOC 设计中的基本原则。

第三章，系统介绍了数字设计中几种常见的数据和控制信号的 CDC 同步电路，总结了这些同步电路在数字 SOC 设计应用中利弊条件。同时也具体分析了常见异步复位信号的同步设计问题，提出了一种在实际项目中常用的异步复位信号同步电路模型。最后通过实际工程项目中常见的验证方法的分析，概括了这些传统的设计方法在 CDC 验证方面所存在的一些缺陷。为后面引出新型的静态验证方法做了准备。

第四章，首先通过静态验证方法的特点分析介绍，对比分析了目前传统验证方法与基于 SpyGlass 的 CDC 静态验证方法的具体优略。继而引入了基于 SpyGlass 的大规模数字 SOC 对 CDC 验证的方案，经过基本验证规则的概述，提出并设计了一套比较完备可行的验证流程。并且针对目前 RTL 设计中一些常见的验证结果，具体分析并对设计缺陷提出了一些建议性的修改方案。通过这种分析总结，加深对这种验证形式和验证方法的理解和应用。

第五章，总结与技术展望。

第二章 数字 SOC 的 CDC 问题分析

2.1 跨时钟域同步概述

从数字系统电路设计的原则出发，数字 SOC 设计完全可以按照统一的时钟信号来进行设计，但是由于功能、时序等诸多客观因素的限制，通常对于一个内部采用统一的时钟来设计的 RTL，为了能够实现更为复杂的系统功能，不可避免地要与周围或者外部的模块电路进行数据之间的交流和通信，而这种通信往往是异步的。按这种设计方式，会把一个系统的电路分布划分成多个时钟区域，每个时钟区域使用一个统一的时钟信号。当数据信号从其中一个时钟域传送到另一个时钟域的时候，对于目的地时钟域设计模块来说，此类信号称为异步输入类型，如果不采取相应的同步设计来防止诸如此类的 CDC 传输，对系统的整体逻辑、时序都会造成影响。

所以，当一个异步信号进入设计模块的时候，首先要确定该异步输入信号所处的稳定电平状态，完成判定读入，系统内部的各个时序、逻辑部分才能协调一致的完成相应的行为动作。而实现这种判定功能的电路，就是信号的同步设计。在数字电路的设计过程中，由于应用场景的不同，其中也会衍生出各种不同作用的同步电路设计。

通过跨时钟域问题的简单介绍分析，发现导致数字 SOC 信号出现不确定性的关键原因是亚稳态问题。如果触发器的 setup time / hold time（建立保持时间）不满足，就可能产生亚稳状态，此时触发器输出端 Q 在有效时钟沿之后比较长的一段时间处于不确定的值（状态），在此期间当中 Q 端为毛刺、振荡、固定的某一电压值，而不是等于数据输入端 D 的值。这段时间被称为决断时间（resolution time）。经过 resolution time 之后 Q 端将稳定到 0 或 1 上，但是最终稳定是 0 还是 1，则是随机发生的，与输入数据没有必然的关系。

一旦触发器发生的亚稳态的情况，由于其输出在稳定下来之前可能是毛刺、振荡、固定的某一电压值，因此推断触发器的亚稳态除了导致逻辑误判之外，输出 0～1 之间的中间电压值还会使下一级接收触发器产生亚稳态（即导致亚稳态的传播）。 其中，逻辑误判有可能通过电路的特殊设计减轻危害（如异步 FIFO 中 Gray 码计数器的作用），而亚稳态的传播则扩大了故障面，难以处理。

2.2 数字设计中常见的 CDC 问题

在目前数字 SOC 工程设计项目中，所存在的 CDC 问题主要有以下几种：

2.2.1 CDC 中的亚稳态传播问题

数字 SOC 中触发器亚稳态值的产生和传输[1][2]模型如图 2.1，当时钟信号 Clk_B 的第二个上升沿到来的时刻，输入数据信号 A 刚好处于变化当中，导致其不满足触发器 F2 的建立-保持时间要求，在输出节点 B 处产生亚稳态值。对于 B 节点产生的亚稳态中间值，后续的三个扇出分支 X1、X2、X3 对其的理解判定则可能是不同的。其中一个扇出可以理解这种亚稳态值为逻辑值 1，与此同时另外的一个扇出也可以将其理解为逻辑值 0，这种后续电路对前级亚稳态的自由理解会在系统电路运转过程中随机的出现。其对于完整的 SOC 系统时序将造成巨大影响。

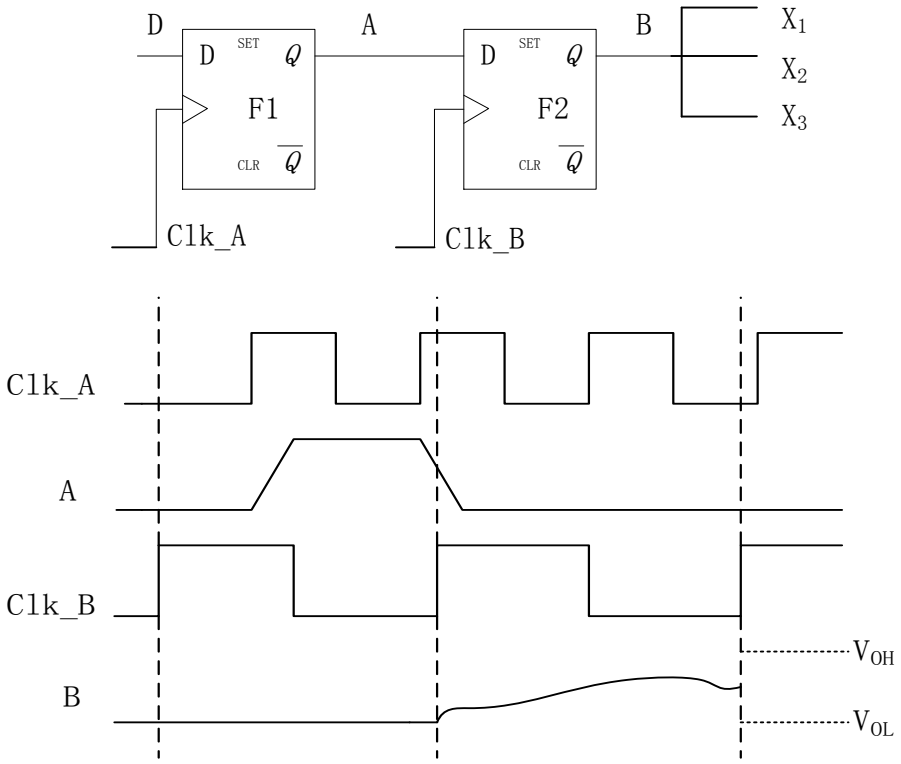


图 2.1 CDC 时候的亚稳态

基于图 2.1 模型，从定性的方面分析，触发器亚稳态产生与传播的概率与异步输入信号翻转的频率、目的触发器时钟翻转频率都呈现正比例的关系。

2.2.2 CDC 中异步输入数据的保持时间问题

当一个数据信号从快时钟域进入慢时钟域的时候，由于该数据信号严格依赖快时钟域信号采样来翻转变化,所以可能会产生与快时钟周期等宽的脉冲型信号。对于慢时钟域的时钟采样边沿来说，是很容易忽略这种形式为短暂脉冲的异步输入数据。造成这种 CDC 传输过程中丢失数据的情况，主要是因为该异步输入数据信号的保持时间不满足目的触发器要求。

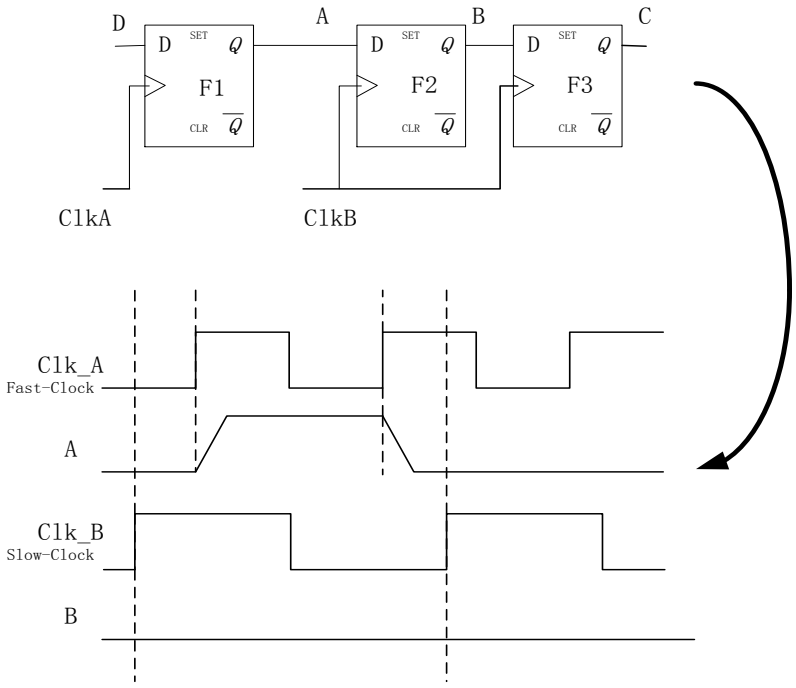


图 2.2 数据保持时间问题电路模型与时序波形

如图 2.2 所示,异步输入数据 A 随着源时钟域信号 clk_A 的一个周期而变化，由于数据 A 的保持时间较短，小于目的时钟域的一个时钟周期，并且两个时钟域的时钟信号之间没有固定的相位关系，结果在 clk_B 的两个上升采样边沿之间，类似脉冲信号的数据信号 A 则被该目的时钟域的同步触发器所忽略掉，造成数据的丢失现象。

2.2.3 CDC 中的数据关联和竞争

图 2.3 中是一个常见的两级触发器型同步设计，其能够起到同步、孤立亚稳态的作用。由于触发器亚稳态的恢复时间不能准确预测，使得此类同步设计存在一个重要特性—Cycle Uncertainty[3]（周期不确定性）。

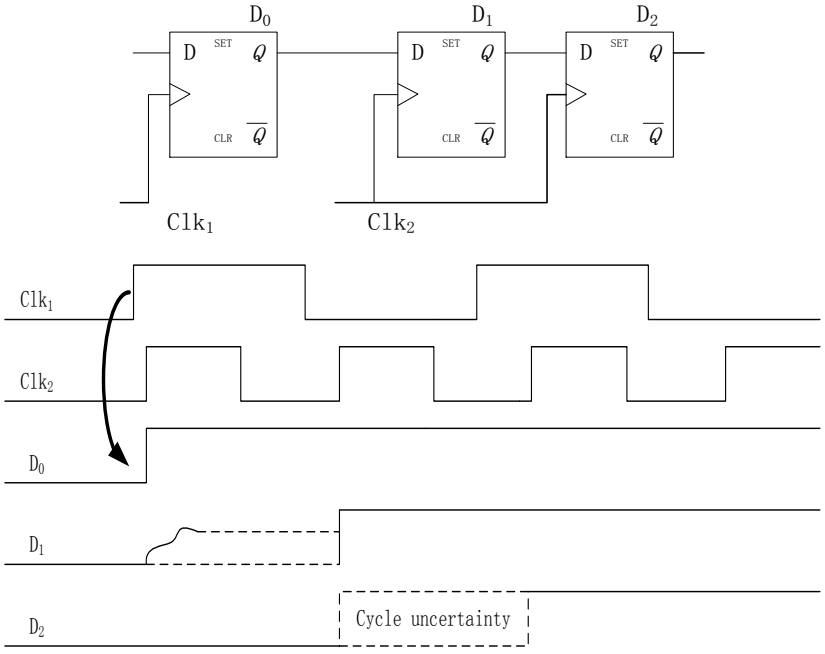
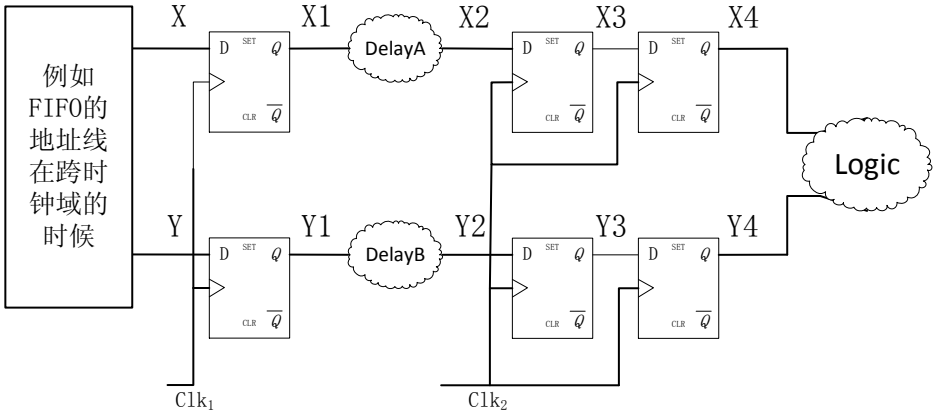


图 2.3 多触发器同步电路模型与时序波形

如图 2.3 时序波形所示，亚稳态虽然被触发器 D_1 所孤立禁止，但对于目的触发器 D_2，在 Clk_2 的第二个时钟上升沿到来时刻，无法确定是否能翻转为稳定值：其有可能在第二个 Clk_2 上升沿时刻翻转为 1；也有可能不翻转，仍然保持 0。直到 Clk_2 的第三个时钟上升沿到来，目的触发器 D_2 才能稳定输出期望值 1。这种情况称为：同步电路的周期不确定性。如果该异步输入数据为单 bit 类型，这种 CDC 问题则不会影响到后续电路的正常功能，但如果为多 bit 数据的情况，采用此种同步必然会造成数据通道传输的紊乱。

基于以上分析，当多 bit 类型的数据信号采用分别同步的情况时，在该信号同时发生跳变之后，由于其每条传输路径上面延迟的不一致，以及同步设计的周期不确定性，很有可能会出现最终输出数据的不一致情况。一旦出现此类问题，对于目的模块逻辑，这将是一组不可预期的数据值，必然引起整体功能的错乱。如图 2.4 模型所示：



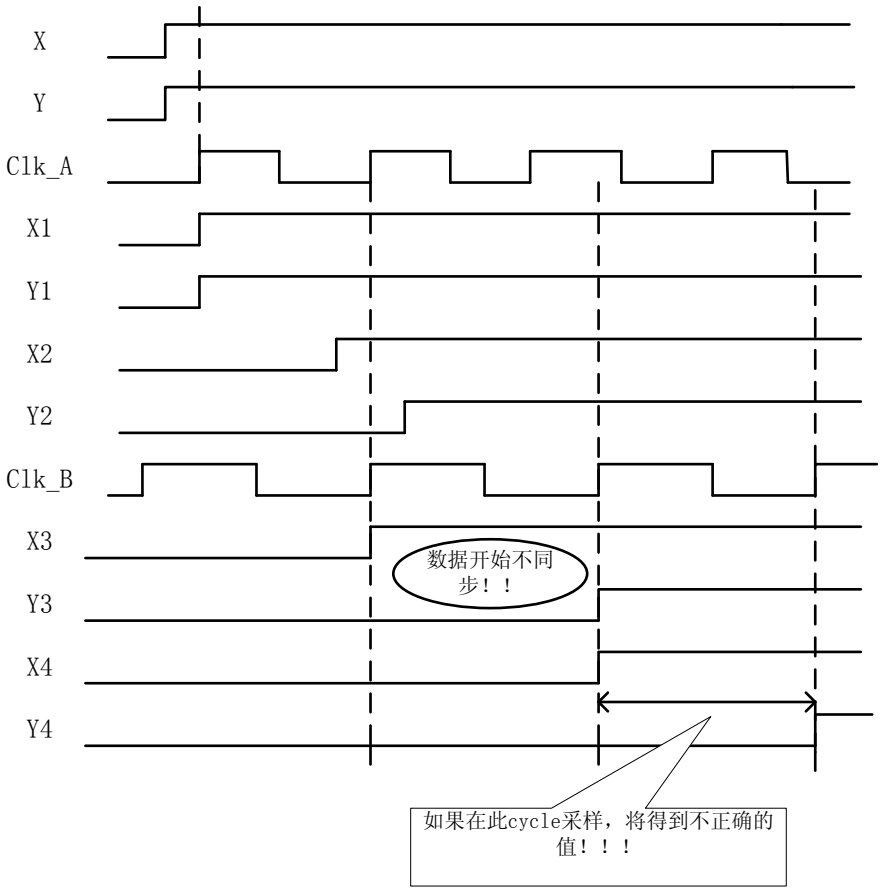


图 2.4 多 bit 信号的汇聚问题

2.2.4 CDC 中复杂的同步设计

对于一些时序要求比较严格的数字 SOC 电路，单纯的使用触发器串联形式进行 CDC 同步已经不可能满足设计要求，必须采取一些更为复杂的同步设计机制。在实际工程项目中，常见的有异步的 FIFO（First In First Out）同步设计、握手协议的同步设计、以及单一使能信号形式的同步设计等等。特别是对于目前大多数基于 ARM 等处理器的数字 SOC 来说，从安全稳定性和传输效率角度考虑，往往会采取所述几种同步设计，来完成 ARM 处理器与高速外设电路之间的 CDC 传输。而对于这种 CDC 同步结构在工作中出现的问题，主要从两个方面来入手，一方面从亚稳态产生和传播，另一个方面是从握手协议等的完善程度。

2.2.4 CDC 中的异步复位同步问题

异步复位在目前时序设计中是最常见的，是指无论时钟信号是否到来，只要复位信号有效，就对系统进行复位。但当复位信号需要释放时刻，却必须依据时钟信号的采样进行。在目前数字 SOC 系统中，由于所容纳触发器数量庞大，物理

上复位信号的布线路径存在很大延迟，所以这种异步复位信号的释放时刻到达芯片内所有触发器的时间点也是不一致的，这样就可能导致系统中一部分触发器处于复位状态，一部分却处于正常的工作状态，引起系统电路时序的不一致。

2.3 亚稳态与 MTBF 分析

在没有进行同步的 CDC 传输过程中，输入的数据信号与该触发器时钟信号成异步关系，其翻转时刻不依赖此时钟信号的有效边沿。而在目的时钟采样边沿时刻附近存在一个 “危险区域”，这个危险的区域通常是该触发器自身的建立时间和保持时间之和。为了能够使目的触发器输出正确的数据信号，异步输入的数据信号翻转时刻不可以落在该时间区域内，即必须满足触发器的最小的建立时间和保持时间之和的最小要求，否则触发器将会产生亚稳态问题[4][5]。

为了解决这种亚稳态的问题，设计中会使用各种不同的同步手段，但是不管采用怎样的方式，总会有同步失效产生亚稳态概率的存在。主要是因为触发器的亚稳态无法在期望的时间范围内得到恢复，并且传递至后续的电路中。设计中定量地衡量：在一定的亚稳态恢复时间 Tr 范围内，触发器仍然出现亚稳态的概率。如公式（2.1）所示：

$$N_{meta}(Tr) = \frac{w}{T_c * T_d} * e^{-\frac{T_r}{\tau}} \qquad (2.1)$$

以上的计算结果为：每秒发生亚稳态错误的个数。由这个模型引出同步设计的平均失效时间[6][7]MTBF（Mean Time Between Failures）定义：一个同步设计的触发器在设计规定的时间内，仍然进入亚稳态的概率。表达式如（2.2）：

$$\mathrm{MTBF(Tr)} = 1/N_{meta}(Tr) = \frac{T_c * T_d}{w} * e^{\frac{T_r}{\tau}} = \frac{e^{\frac{T_r}{\tau}}}{w * f_c * f_d} \qquad (2.2)$$

以上可以看出，在这样的一个模型中，MTBF 对于数据翻转频率f_d、目的时钟域等的频率f_c、敏感时间窗口 w、亚稳态恢复时间Tr以及恢复时间常数τ都是密切相关的。尤其是对于恢复时间常数τ。

在如今多处理核、高速低功耗数字 SOC 设计中，基于以上同步设计 MTBF 数学模型，可以分析同步触发器 MTBF 随时钟频率和工作电压的具体变化情况，得出实施同步设计时一些基本注意规则。

2.3.1 基于不同时钟域的 MTBF 分析

在早期的 SOC 设计中，通常只是提供一个公共的时钟信号给所有系统模块，包括各种不同的 IP。对于这样的设计结构，是不需要采取特殊的同步设计来对异步输入数据信号进行相应保护的，因此也就不存在以上所述的一系列 CDC 问题，

对于这种单时钟信号的系统，往往将其称为同步的系统(当然可能也会由于时钟树的庞大所产生的触发器的不同步问题，对于此种情况在此不作讨论)。如图 2.5 所示：

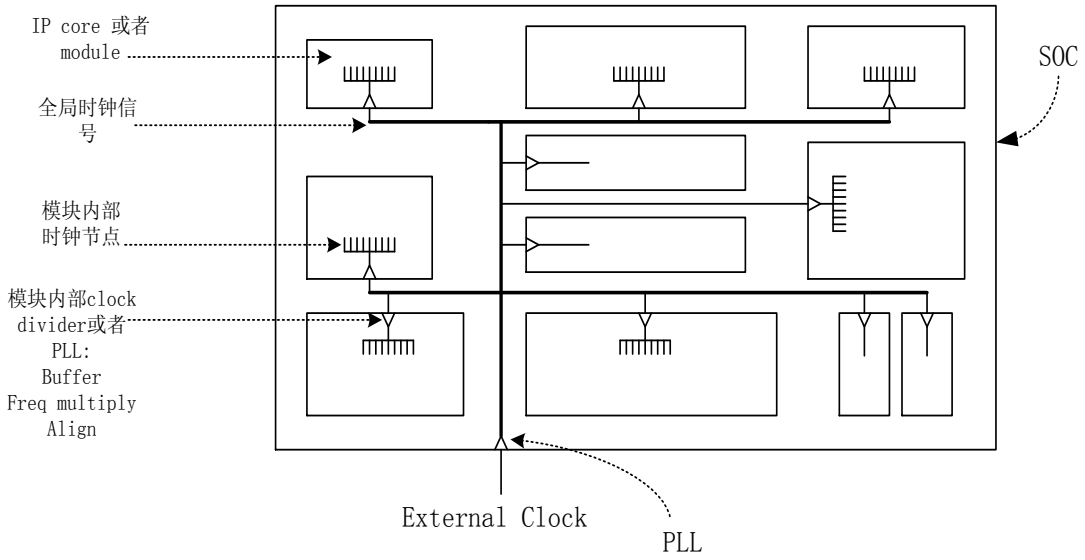


图 2.5 单时钟域 SOC

由上图可以看出，在单时钟域 SOC 中，全局时钟信号同时抵达各个 IP 或者功能模块。这样保证数据安全的从一个模块向另外一个模块的传输通信。对于这样的 SOC 架构，只需要在设计的过程中，通过添加时钟树平衡器或者缓冲器等器件保持时钟树传递的平衡即可。

对于上述的单时钟域数字 SOC 设计结构，分析相关的 MTBF 模型其实是不必要的。由于在系统中的每个寄存器上的时钟信号相位都是相同的。所以从理论上讲，数据 CDC 传输之间不会存在亚稳态等问题。

而在当今的大规模数字 SOC 设计中，能够碰到的单时钟域的情况是极少数。对于绝大多数的数字系统电路设计来说，必须要对系统分配多个 clock-domain。如图 2.6 所示的一个多时钟域 SOC 的模型结构。

分配和划分多个时钟区域，并不是真正的将这些单个独立的 clock-domain 彼此之间从物理上相互分割开来，而只是在概念上定义的一个时钟区域的划分。而且在实际的数字 SOC 中，这些不同的时钟区域也有可能会是相互重叠的。例如：一个模块内部的某个部分和另一个模块内部的某块电路可能同属于一个时钟区域。而客观上也不太可能在物理上进行明确的分离。

由图2.6不同传输路径看出，如果在一个独立时钟域内部进行的数据的传输，在进行相应的 CDC 同步设计与验证的时候，是不进行特别检查的。可以等同于如上所述的单时钟域 SOC 的处理方式。但是对于不同时钟域之间的数据传输或者从 SOC 外部输入的数据，这种情况则需要考虑同步设计的处理方式以及相关的

CDC 验证检查等。

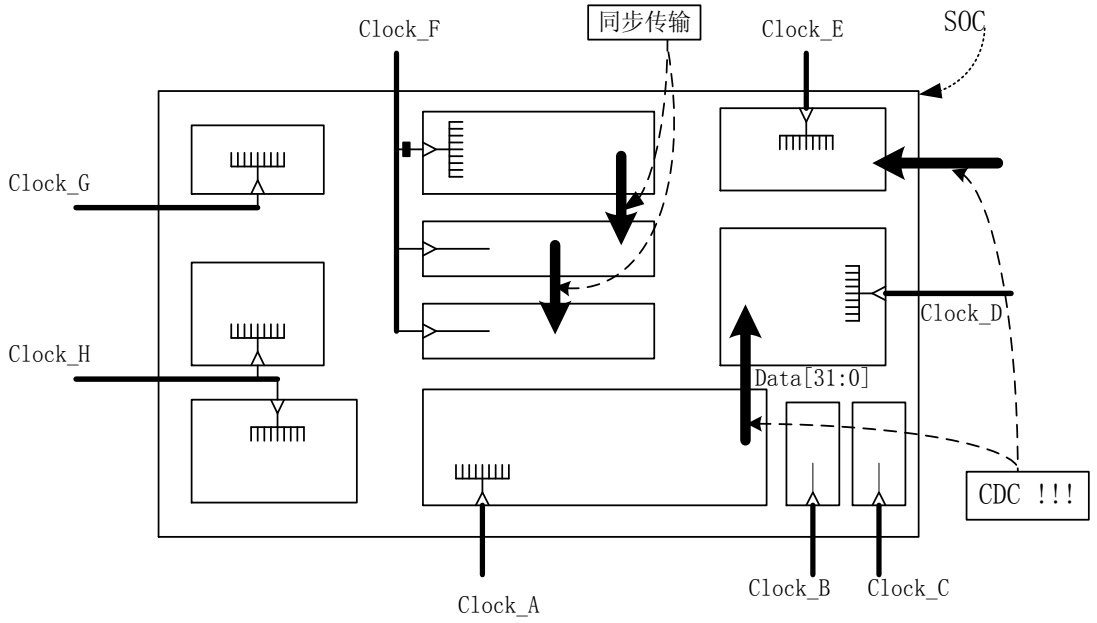


图 2.6 多时钟域 SOC

对于设计中出现和应用越来越多的 clock-domain，其主要是因为一下几点的原因：

(1) 系统或者协议本身对多时钟架构的需求：

例如 SOC 和外部进行相关的通信传输时候，其需求两种以上的时钟频率。

(2) SOC 的规模越来越大：

随着规模的不断增大，可以看到一些 clock-skew（时钟扭曲）问题也随着系统内部的触发器数量的不断增多而愈发突出。迫使设计者不得不进行多时钟域的设计。

(3) 设计的复杂化：

例如，如果系统中有一个 16 位的 20MHz 处理器、1 位的 100MHz 串口以及 1MHz 的 IO 控制电路。(a)，首先不可能将他们三者都优化在 100MHz，尽管那样显得会更加的简单和便宜。(b)，其次同样也没有必要将所有的电路都设计在 1MHz 下运行，尽管这样可能会有比较好的稳定性。但是，这些显然是不符合如今的应用和设计需求的。

(4) 基于功率消耗的考虑：

首先，应该知道的是，开关频率是和动态的功耗是成正比的；其次，如今各种不同的 IP 设计的都能够在更低的电压下工作；所以在低功耗的需求下，更加迫切的要求多时钟架构设计的增多。例如，一款简单的数字基带芯片，由于其包含了很多的应用模块和控制模块，比如 CGU、SCU、2G_DSP、3G 等等。不同的模块则需求多个不同频率的时钟信号，如果这些模块之间的数据发生 CDC 传

输，必然要考虑同步。以下是 3G 模块部分时钟信号，可以看出目前数字芯片时钟信号的基本规模：

表 2.1 3G 模块部分时钟信号

Block instance short name	Block clock names	CGU clock names	Master clock
3G_SUBSYSTEM	clk_26m	clk_26m	mclk_26m
3G_SUBSYSTEM	clk_26m_age	clk_26m_age	mclk_26m
3G_SUBSYSTEM	clk_32k	clk_32k	PIN
3G_SUBSYSTEM	clk_3g_124m	clk_3g_124m	mclk_3g_124m
3G_SUBSYSTEM	clk_3g_178m	clk_3g_178m	mclk_3g_178m
3G_SUBSYSTEM	clk_3g_208m	clk_3g_208m	mclk_3g_208m
3G_SUBSYSTEM	clk_3g_dsp	clk_3g_dsp	mclk_3g_dsp
3G_SUBSYSTEM	clk_3g_hsorx	clk_3g_hsorx	mclk_3g_hsorx
3G_SUBSYSTEM	clk_3g_searcher	clk_3g_searcher	mclk_3g_searcher
3G_SUBSYSTEM	clk_3g_system	clk_3g_system	mclk_3g_system
3G_SUBSYSTEM	clk_3g_teak	clk_3g_teak	mclk_3g_teak
3G_SUBSYSTEM	clk_ahb_L1	clk_ahb_3G	mclk_ahb_macom
3G_SUBSYSTEM	clk_lte_top	clk_lte_top_PSD_G3	mclk_lte_top
3G_SUBSYSTEM	ref-clk	clk_26s_gated	clk_26s
3GSYS_TRACE_IF	clk	clk_systrace_PSD_G3	mclk_systrace

为了消除这些多时钟域 SOC 的 CDC 异步传输问题，通常的同步设计做法是用触发器串联的形式来增加触发器的亚稳态恢复时间T_r，下面就来通过具体的电路模型分析一下一级、两级、三级触发器串联的亚稳态恢复时间的模型，以及各级触发器串联同步设计的 MTBF 分析：

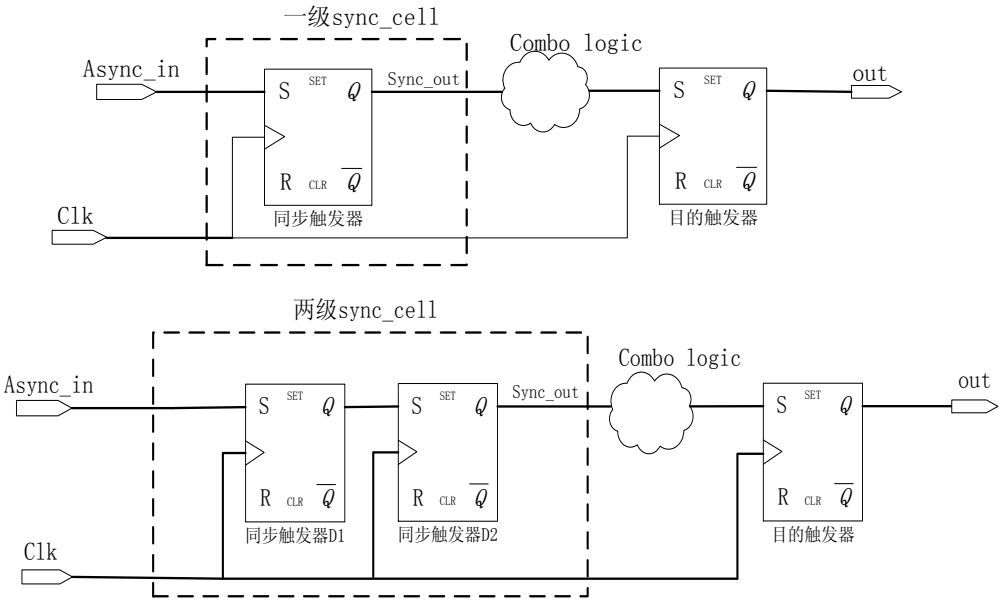


图 2.7 CDC 同步设计电路模型

由图 2.7 中电路模型，对于一级触发器同步设计，必须要考虑后续的的组合

逻辑的延迟时间和目的触发器的建立时间，所以实际的模型应该为：$T_r = T_c - (T_{combo} + T_{setup})$。可以看到，一级触发器的亚稳态恢复时间不仅取决于目的时钟的周期，而且还与中间级的组合逻辑延迟有关。

同理，对于两级触发器串联的同步设计，由于插入的一个额外的触发器，使得两级触发器同步设计中的触发器 D2 的亚稳态恢复时间模型应该变为：$\mathrm{T} = T_c - T_{setup}$。主要是因为两级触发器串联的同步设计中间没有组合逻辑。以此类推，三级模型应该为$\mathrm{T} = 2T_c - T_{setup}$；四级模型为$\mathrm{T} = 3T_c - T_{setup}$。由此可以看出：在两级以上的触发器串联形式的同步设计中，对于 sync_cell 的最后一级触发器的亚稳态恢复时间来说，只与自身的触发器建立时间有关。

基于上面的亚稳态恢复时间分析，可以带入得出多时钟域 SOC 对不同时钟频率的 MTBF 值。由上述的(2.2)式，参考如下的设计实例，目的时钟域的频率f_c，异步输入数据的翻转频率$f_d = 10\% * f_c$，时间常数$\tau = 50\mathrm{ps}$，敏感时间窗口$\mathrm{w} = 0.05\mathrm{ns}$，组合逻辑的延迟时间为$T_{\mathrm{combo}} = 0.01\mathrm{ns}$，触发器的建立时间为$T_{\mathrm{setup}} = 0.02\mathrm{ns}$。

一级触发器、两级触发器以及三级触发器的亚稳态恢复时间T_r如表 2.2 所示：

表 2.2 触发器的T_r与目的时钟频率

目的时钟域 时钟的频率f_c	目的时钟域 时钟周期T_c/ns	一级触发器 T_r/ns	二级触发器 T_r/ns	三级触发器 T_r/ns
32KHz	31250	31249.97	31249.98	62499.98
26MHz	38.46154	38.43154	38.44154	76.90308
52MHz	19.23077	19.20077	19.21077	38.44154
108MHz	9.259259	9.229259	9.239259	18.498518
256MHz	3.90625	3.87625	3.88625	7.7925
480MHz	2.083333	2.053333	2.063333	4.146666
800MHz	1.25	1.22	1.23	2.48
1000MHz	1	0.97	0.98	1.98
1200MHz	0.833333	0.803333	0.813333	1.646666
1500MHz	0.666667	0.636667	0.646667	1.313334

根据以上表中不同级串联触发器的亚稳态恢复时间T_r，其对应的 MTBF 计算值如下：

表 2.3 MTBF 随目的时钟频率f_c变化

目的时钟域 时钟频率f_c	目的时钟域 时钟周期T_c/ns	一级触发器 MTBF/year	二级触发器 MTBF/year	三级触发器 MTBF/year
32KHz	31250	$+\infty$	$+\infty$	$+\infty$
26MHz	38.46154	$+\infty$	$+\infty$	$+\infty$
52MHz	19.23077	$1.3995 * 10^{154}$	$1.709 * 10^{154}$	$+\infty$
108MHz	9.259259	$7.935 * 10^{67}$	$9.695 * 10^{67}$	$2.579 * 10^{148}$
256MHz	3.90625	$4.513 * 10^{20}$	$5.512 * 10^{20}$	$4.683 * 10^{54}$
480MHz	2.083333	18826.2839	22994.4751	$2.866 * 10^{22}$
800MHz	1.25	$3.916 * 10^{-4}$	$4.783 * 10^{-4}$	$3.444 * 10^{7}$
1000MHz	1	$1.689 * 10^{-6}$	$2.0625 * 10^{-6}$	1000.656
1200MHz	0.833333	$4.183 * 10^{-8}$	$5.1095 * 10^{-8}$	0.88434
1500MHz	0.666667	$9.551 * 10^{-10}$	$1.1666 * 10^{-9}$	$7.203 * 10^{-4}$

可以看出，当目的时钟域的采样频率越来越高的时候，如果采用一级触发器的同步设计，则完全不能满足 MTBF 的设计要求，特别是在目的时钟域高达 800MHz 的时候，出现亚稳态错误的几率则变得非常高，约为平均每 3.43 小时就将出错一次。若在两级触发器的串联同步设计情况下，可以看到这样的情况似乎也并没有得到很好解决，依然存在很高的出错几率。尽管在三级触发器的时候，这样的问题得到了很大程度的缓解，例如：MTBF 在 800MHz 的情况下，已经是$3.444 * 10^{7}$年，这样是非常稳定的。但是同样会发现，随着时钟频率的不断提高，这样的同步失效的情况永远都会存在。

尽管这样的触发器串联的方式在实际的设计中非常的常见。但是通过上述的一系列亚稳态 MTBF 随时钟频率的分析，可以看出不同级数同步触发器的 MTBF 都与对应的时钟频率成反比的关系。同时也应该清楚，这个 MTBF 对时间常数τ也相当敏感，不同芯片之间的τ值也是不尽相同的，即使是一个芯片，其随着温度等周围环境因素的不同，τ 值也会在很大范围内浮动，因此这样也就会间接地导致系统中同步设计的 MTBF 在很大程度范围内变化，所以对于τ值的这种不确定的浮动，也应该加以必要的考虑。

在此可以总结出一个基本的结论：对于高速数字 SOC 设计来说，随着目的时钟域的时钟频率越来越高，最常见的两级触发器串联的形式已经不能够满足系统

对 MTBF 的要求。或许可以应用三级触发器的形式，但是这样势必会增大系统的总体延迟。总体上这样串联触发器的形式似乎也不能够彻底的解决这种越来越高速的 CDC 问题，这就迫使设计者找到更加合理的同步方法。

2.3.2 基于不同电压域的 MTBF 分析

由于 MTBF 的大小与时间常数τ有着十分密切的关系，只要τ值发生细微的变化，同步设计触发器的 MTBF 则将会在很大的范围内变动。参考两级串联触发器同步设计，$f_c = 200\text{MHz}$，异步输入数据$f_d = 10\% * f_c$， $\text{w} = 0.05\text{ns}$，组合逻辑的延迟为$T_{\text{combo}} = 0.01\text{ns}$，触发器的$T_{\text{setup}} = 0.02\text{ns}$。

表 2.4 时间常数τ对 MTBF 的影响

两级触发器串联同步设计模型下：	MTBF （单位：/年）
时间常数τ = 50ps	$2.8 * 10^{30}$
时间常数τ = 70ps	$1.25 * 10^{18}$
时间常数τ = 90ps	$1.7 * 10^{11}$
时间常数τ = 110ps	$7.275 * 10^{6}$
时间常数τ = 130ps	6870.41
时间常数τ = 150ps	41.5662
时间常数τ = 160ps	5.2189
时间常数τ = 165ps	2.03217
时间常数τ = 170ps	0.83644

以上的数据结果可以看出，随着 t 值的细微增加，触发器 MTBF 从极其稳定变为每 0.83644 年出现一次错误。

对于亚稳态的时间常数τ，除了环境温度因素，其还受到集成电路工艺技术相关的诸多因素的影响。包括晶体管的速度，特别是工作电压。随着日益提高的集成电路工艺技术水平，能够在单位面积上集成的晶体管的数量也在急剧的增多，同时芯片的工作电压也是不断的降低。这也使得系统电路中触发器的时间常数τ也发生着巨大变化。

对于一个 CMOS 晶体管，从定性的角度来分析，其自身的跨导g_m是和电压V_{DD}成如下的比例关系的：

$$g_m \propto \frac{I_D}{\gamma * U_t} \tag{2.3}$$

其中γ为一个非线性因子，$U_t = k * t/q$，因此可以看到电导和漏电流是成正比例关系的。而漏电流I_D也与工作电压 VDD 成这样的相关的关系：

$$I_D \propto \beta(V_{DD} - V_{thre})^{\alpha} \qquad (2.4)$$

式中β为一个非线性因子，同时α也为一个与晶体管宽长比、电迁移率相关的一个因数，V_{thre}为晶体管的阈值电压。所以可以看出，电导和工作电压成α次方的关系。

而时间常数又和晶体管的电导参数成反比例关系：

$$\tau \propto \frac{\beta}{g_m} \qquad (2.5)$$

式中的β也是一个非线性因子。由此可以推导出时间常数和工作电压的比例关系：

$$\tau \propto (V_{DD} - V_{thre})^{\alpha} \qquad (2.6)$$

由式（2.6）可以看出，随着工作电压的不断的减小，时间常数将会是急剧的增加。分析系统工作电压与时间常数的关系可知：如果工作电压不断地减小，相应漏电流也会降低，这样将导致一旦晶体管进入到亚稳态，其脱离亚稳态的时间也就会增加。因此时间常数是关于触发器等脱离亚稳态的时间长短的一个衡量标准。下面就可以具体的分析低功耗数字 SOC 设计中的 power-domain 不同对于亚稳态 MTBF 的影响[8][9]。

在如今的大规模 SOC 设计中，最关键的一个设计目标之一就是低功耗，而 clock-domain 的划分种类繁杂，目的都是为了允许数字系统中的不同模块，能够工作在不同的电压下。图 2.8 为 power-domain 在低功耗 SOC 的几种具体实现形式：

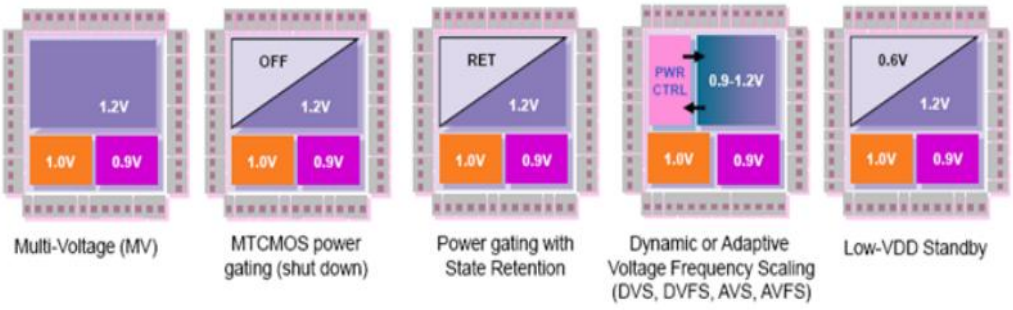


图 2.8 SOC 设计中的 power-domain 划分类型

为了低功耗的要求，划分不同的 power-domain，并且使用更低的工作电压，但是晶体管的阈值电压并不会随着工作电压的降低也成比例的降低，当一个触发器进入到亚稳态的时候，它的电压接近正常工作电压的一半。随着正常工作电压的降低，电路中亚稳态的电压也就非常的接近于晶体管的阈值电压，当这两个电压大小非常接近的时候，相应的晶体管的增益会不断的降低并且触发器从亚稳态中传输的时间也会增长。虽然设计者在不断的追求更高速度的晶体管，但是随着工艺的提升和工作电压的不断下降，这种阈值电压的顾虑很显然比一味的提高晶

体管的速度更加重要。否则，触发器的 MTBF 将会随着愈来愈低的工作电压，而变得越来越糟糕。

由此两种基于不同 clock-domain 和不同 power-domain 的同步亚稳态的 MTBF 的分析，可以总结得出：对于一般的工程项目，时钟的分布频率不是非常高的时候，这种情况下采取两级触发器串联的形式即可获得比较好的效果；但是对于时钟频率较高的设计场合，还是需要考虑使用更多级数的触发器串联形式或者采取能够提供更长亚稳态恢复时间的先进同步设计。而对于越来越低的工作电压所导致的同步亚稳态 MTBF 同时降低的情况，数字设计也应该采取相对稳定的同步设计电路，从而避免由于电路中 power-domain 划分和工作电压下降所造成的失效问题。

第三章 CDC 同步设计分析与传统验证方法

基于前面 CDC 传输问题概述、以及触发器产生亚稳态和不同设计架构的 MTBF 具体分析，在这一章，将重点介绍和分析一下在数字 SOC 设计中经常能碰到的一些基本的同步电路的设计，并结合当今高速 SOC 系统的具体设计要求，对其应用场合优势和限制条件做出一系列的分析总结。同时给出设计流程中传统验证方法在 CDC 同步检查方面的缺陷分析，为后续章节新的验证方法引入做预先准备。

3.1 单 Bit 信号 CDC 同步设计分析

有上一章同步设计亚稳态 MTBF 的分析可以得知，只要提供足够的亚稳态的恢复时间，触发器产生亚稳态的概率就会极大的降低。这种增加亚稳态恢复时间的传统方法就是通过增加触发器串联级数来完成的。

首先在数字系统设计过程中，发生这种 CDC 数据传输的信号类型有很多种，其中单 bit 信号类型的 CDC 同步设计电路分析分为以下具体两种传输类型。

3.1.1 慢时钟域到快时钟域的同步情况

对于这种从慢时钟域到快时钟域的 CDC 传输情况，同步设计分为两类：一类为对电平敏感的同步设计，另一类则是以边沿敏感的同步设计。

一、电平同步设计

电平相关的同步设计核心是一个简单的触发器串联，根据时钟运行速度的差异，常使用的方式有两级触发器串联或者三级触发器串联等两种形式，图 3.1 是一个两级的串联形式同步设计：

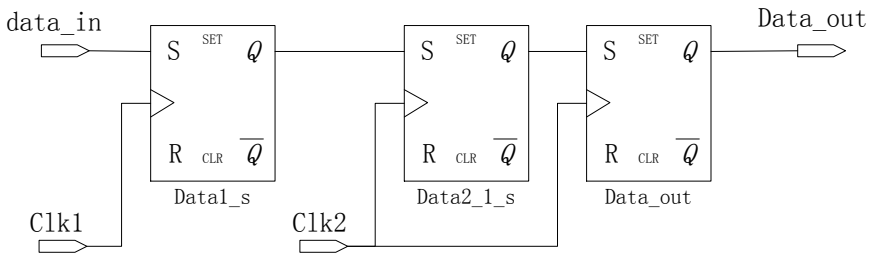


图 3.1 电平同步设计电路结构

其对应的 ModelSim 仿真模型如图 3.2 所示：

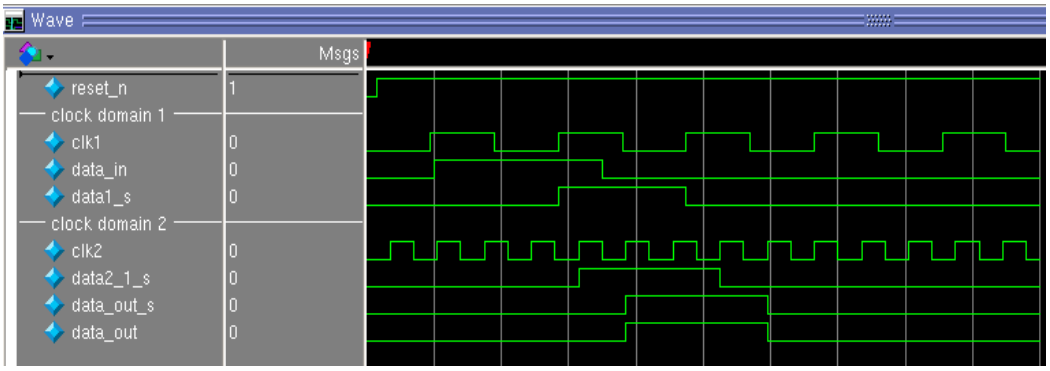


图 3.2 电平同步设计仿真波形

目的时钟域的时钟频率是源时钟域频率两倍以上，该电路同步工作正常；一旦目的时钟域频率达不到这样的要求的话，异步信号的保持时间的问题就会凸显出来。

二、边沿同步设计

由于其同步的对象为异步信号的边沿，因此也分为上升沿同步电路和下降沿同步电路，但同步核心与前面所述电平同步设计的原理一致。不过在此前同步电路基础上添加一些逻辑操作，便可完成捕捉源时钟域数据信号的一个上升沿，同时在目的时钟域产生与时钟周期等宽、高电平有效的脉冲。其基本的电路结构以及仿真波形如下：

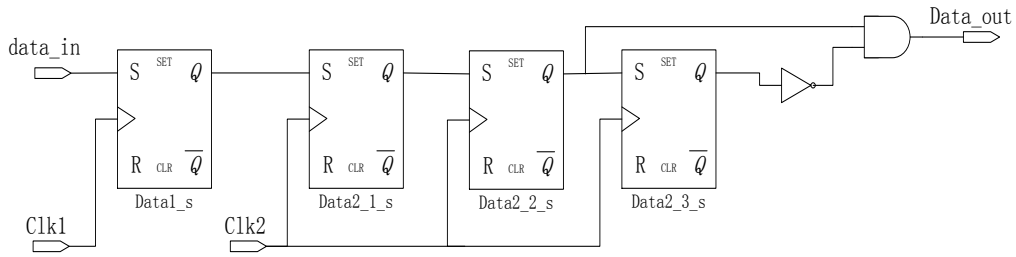


图 3.3 边沿同步设计电路结构

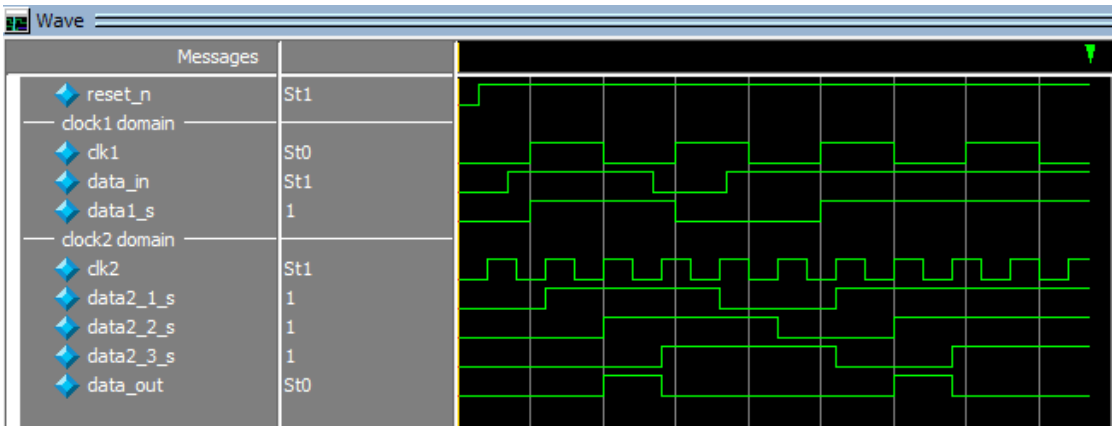


图 3.4 边沿同步设计仿真波形

以上可以看出，边沿同步设计捕捉到了异步输入数据的两个上升沿，同时在

data_out 端输出了两个域目的时钟域时钟 clk2 周期等宽的脉冲。但是这种成功的捕捉是有条件的，一旦目的时钟域频率小于源时钟域频率，或者异步输入信号保持时间不够，都会引起捕捉的失败和同步的失效。

从边沿同步设计的电路结构，如果在结构上稍稍做一些调整，则就能将其变换成另外一种功能的同步设计电路。比如：将与门的两个输入端交换使用，就可以构成一个检测输入信号下降沿的同步设计；将与门改为与非门可以构建成一个产生低电平有效脉冲的电路。

因此，电平和边沿同步设计的优缺点：

1) 触发器串联的形式势必会增加系统的总体延迟。

2) 对于两级触发器串联的同步设计，虽然能够保证其在一定的时钟频率范围内很好的同步，但是如果对于时钟频率的继续高速提升，这种同步设计显然是不能满足设计要求的。

对于这种类型的同步设计，其对输入信号的要求都是比较严格的。异步输入的信号必须满足一定的保持时间。这一点是十分关键的。

3.1.2 快时钟域到慢时钟域的同步情况

上面主要分析了从慢时钟域到快时钟域的同步设计的几种情况，下面则具体分析当单 bit 的异步数据信号从快时钟域进入慢时钟域的时候，数字 SOC 中常见的同步设计电路类型。

一、 无反馈脉冲同步设计

其基本的功能是从某个时钟域取出一个单时钟周期宽度的脉冲，然后在目的时钟域建立另一个单时钟周期宽度的脉冲信号。电路模型如下所示：

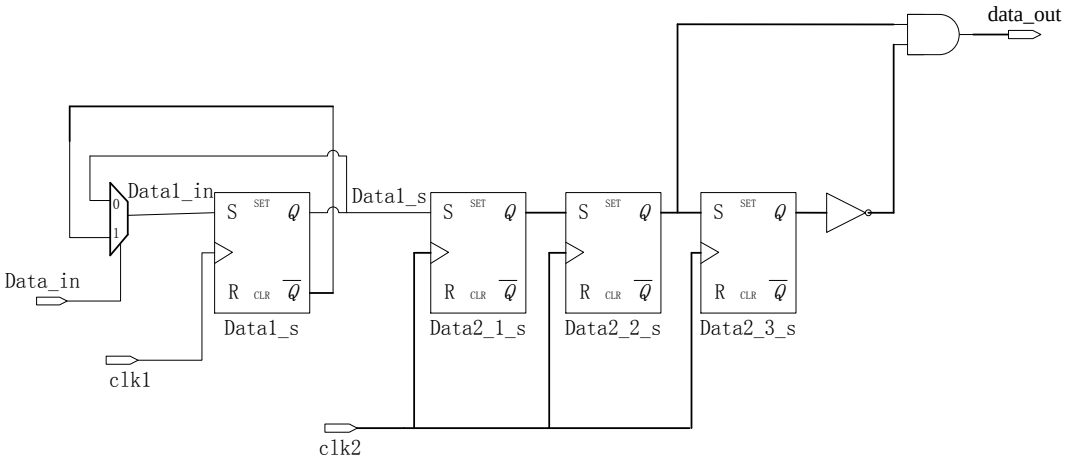


图 3.5 无反馈脉冲同步设计电路结构

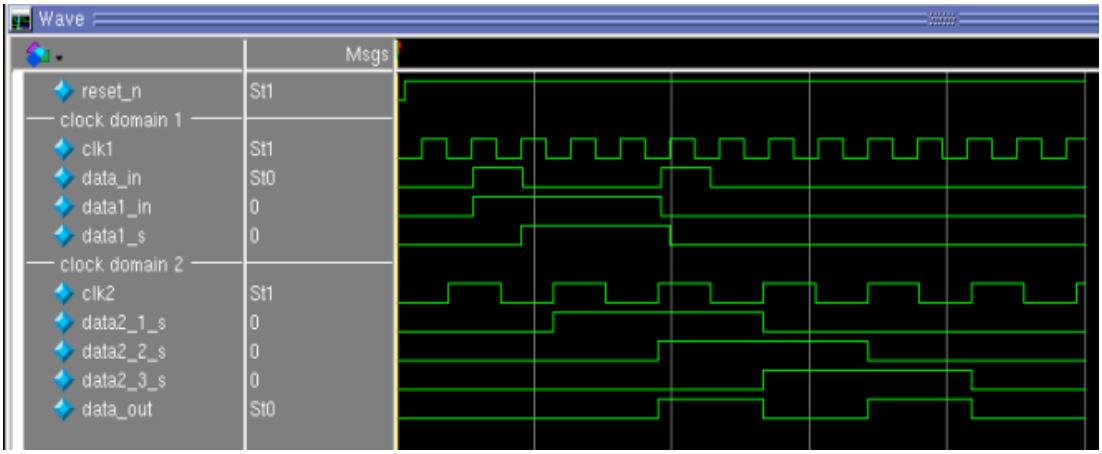


图 3.6 无反馈脉冲同步设计仿真波形

由图 3.6 仿真波形可以看出，当输入的脉冲信号在源时钟域经过一个采样之后，会形成一个比较宽的信号，然后送给目的时钟域，经过一个采样和异或的处理之后，在目的时钟域得到了两个和目的时钟域周期等宽的脉冲信号。

基于上面的电路结构以及仿真结果可以总结出：这种无反馈的脉冲同步设计还是有一个限制：那就是异步输入的脉冲信号的间隔问题，如果相邻两个脉冲之间的间隔时间小于一个目的时钟域时钟周期，那么有可能会出现目的时钟域采样不到的现象；如果相邻两个脉冲之间间隔时间大于一个目的时钟域时钟周期但是小于两个这样的周期，那么有可能在目的时钟域产生的脉冲是一个连续两个时钟周期的宽度，这将导致电路不能区分脉冲信号；如果相邻两个脉冲之间的间隔时间大于两个目的时钟域时钟周期，这样的同步设计将会获得最理想的输出结果。为了解决这些限制问题，引入带 busy 反馈脉冲同步设计。

二、带 busy 反馈脉冲同步设计

通过在上述无反馈脉冲同步设计的 data2_2_s 的输出添加反馈信号至源时钟域，在经过在源时钟域的同步并且进行上升沿采样，从而能够提供给源时钟域一个脉冲信号，其通过一个上升沿采样的处理电路，产生一个相应的标志位信号 pulse_edge_flag，表明要开始一个数据线上的传输动作；同时当数据传输完成的时候，会从目的时钟域同步回来一个反馈的信号，经过一个下降沿采样的处理电路时，产生一个相应的标志信号 feedb_edge_flag，表示数据线上完成了当前的传输。在源时钟域里面同时送给所专门设计的 busy_fsm 模块，通过这个状态机模块可以得到一个 busy_o 信号，用来表明目的时钟域是否能够接受下一个数据。当源时钟域的数据源得到这样的一个信号的时候，就很方便的知道当前传输的数据所处的状态，同时也能知道在什么时候可以开始下一次新数据的传输发送，通过下面的这个电路模型来表示这样的一个带 busy 反馈型的同步设计的具体传输条件：

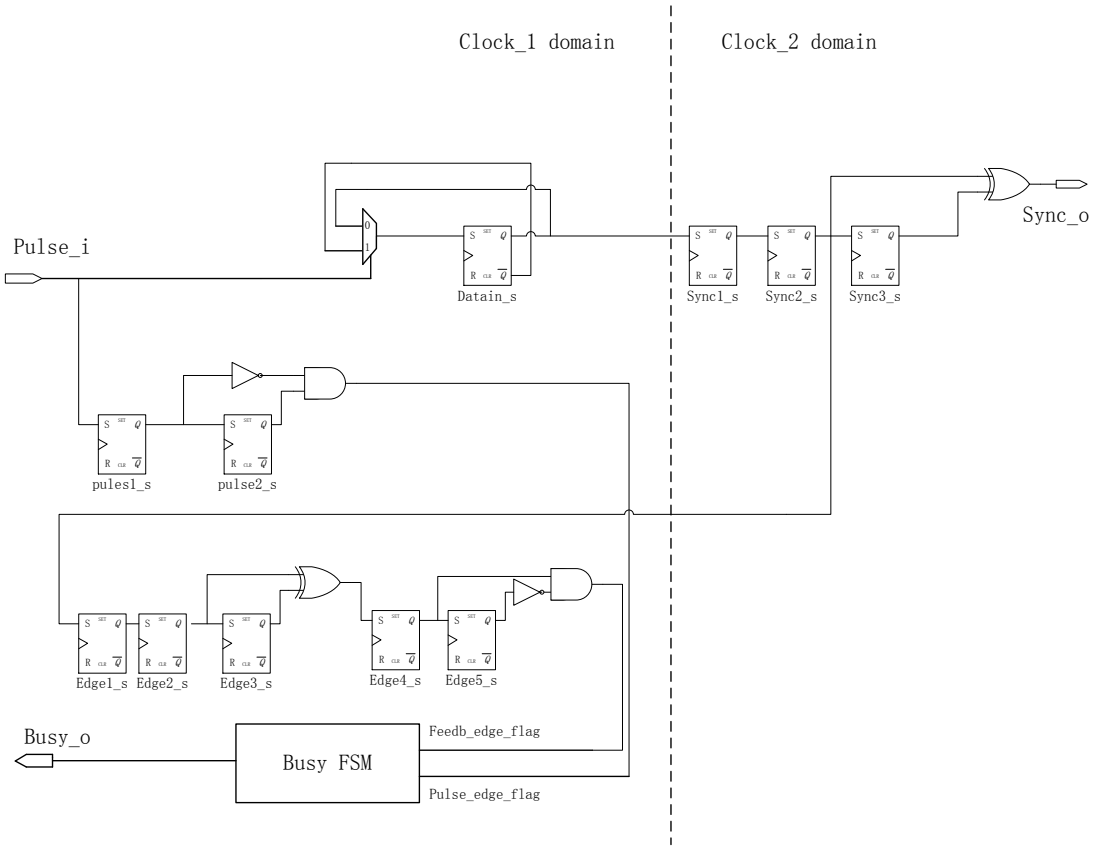


图 3.7 带 busy 反馈型脉冲同步设计电路结构

由模型可知，当输出信号 busy_o 为 0 时，表示目的时钟域可以接受下一次的数据输入，当 busy_o 为 1 时，表明目的时钟域不适合接受新的异步输入脉冲。

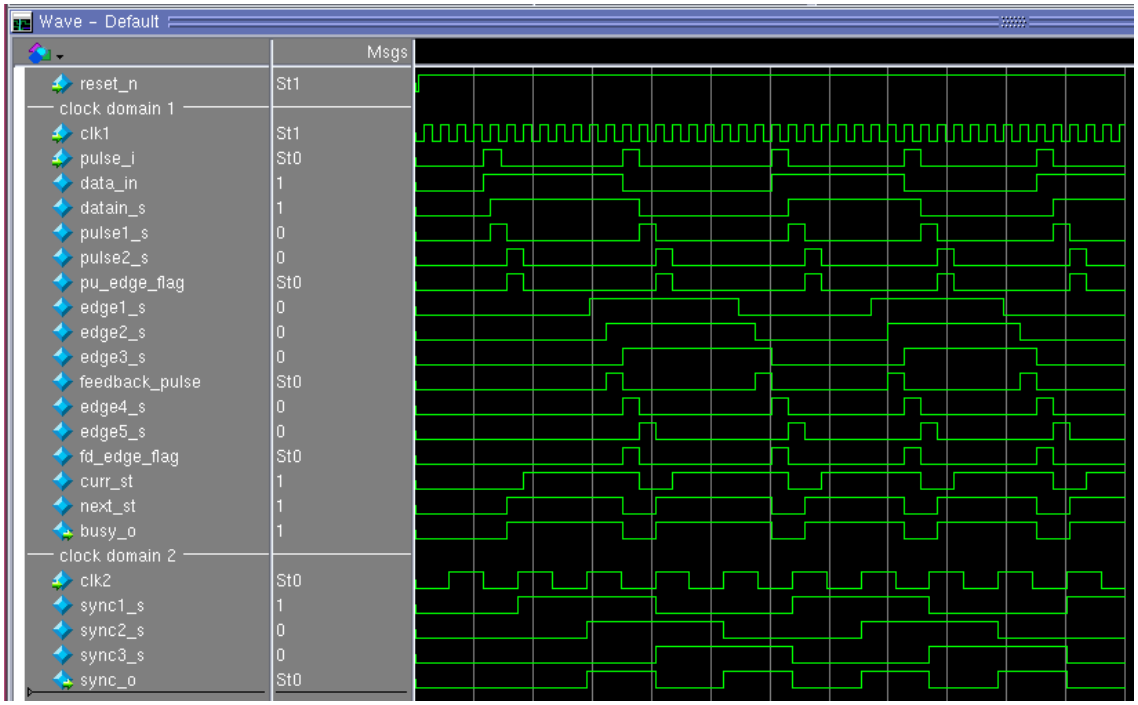


图 3.8 带 busy 反馈脉冲同步设计仿真波形

该设计保证了相邻两个脉冲输入信号间隔时间的问题。特别是对于源时钟域频率

远高于目的时钟域的时候，但电路的设计却相对比较复杂。

3.2 多 Bit 信号 CDC 同步设计分析

以上分析了几种单 bit 信号的 CDC 同步设计，然而在实际应用场合中，跨时钟域传送的不只是这些简单的单 bit 信号，数据总线、地址总线以及相关的控制总线也一样要经过 CDC 到达接收模块。如果单纯的去同步每一个 bit 位上面的信号，则会出现 CDC 中的数据汇聚问题，造成输出数据的不一致。为了避免这样一系列问题的出现，设计者必须采用一些特定的设计来完成多 bit 信号的 CDC 同步传输。

3.2.1 握手协议同步设计

当需要对上述的一些总线信号进行 CDC 传输的时候，如果两个时钟域之间不能够预知相互的响应时间，握手协议的同步设计[10][11]是在 SOC 设计中经常采用的解决方案。如图 3.9 所示电路模型：

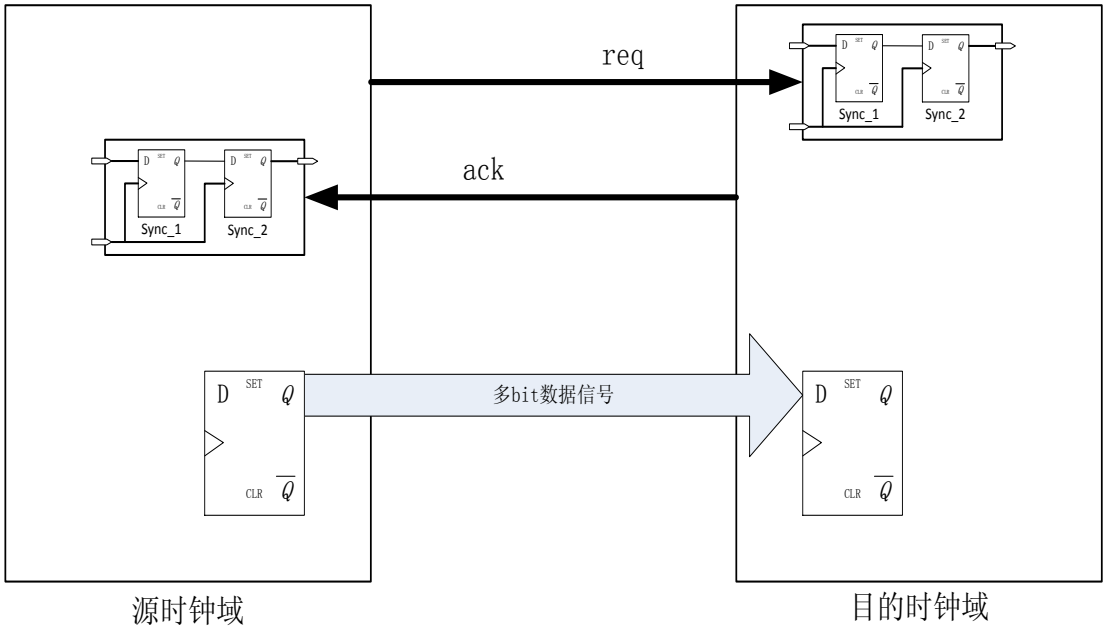


图 3.9 握手协议同步设计结构图

通常会根据实际应用中的需求，将握手协议的同步设计分为两种形式：全握手的同步设计（Full-Handshake）和部分握手的同步设计（Partial-Handshake）。全握手时序如图 3.10 所示：

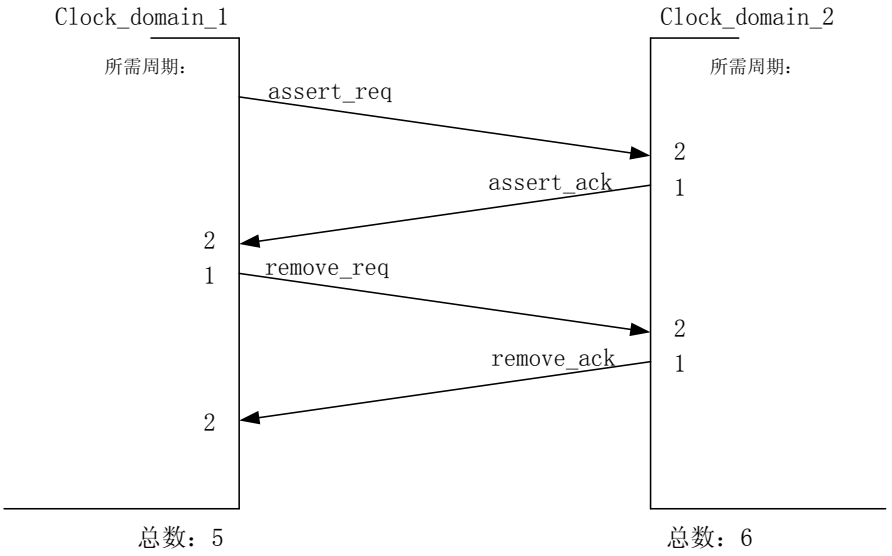


图 3.10 全握手协议次序

对于全握手时序，传输一个多 bit 数据信号时，在源时钟域需要耗费 5 个时钟周期;在目的时钟域则需要耗费 6 个时钟周期。其适应性会非常强，因为 req-ack 信号全部是交替响应，使得双方的时钟域都能够非常清楚对方在每个时刻的传输状态。但同时也可以看出完成一个数据的传输所消耗的时钟周期比较长，使得多 bit 数据 CDC 传输的速度较慢。

要进行高速 SOC 系统设计，必须要对这样的握手协议进行一定的优化处理，尽量的提高多 bit 数据在 CDC 传输过程中的速率。第一种简化的部分握手协议同步设计的传输时序如下：

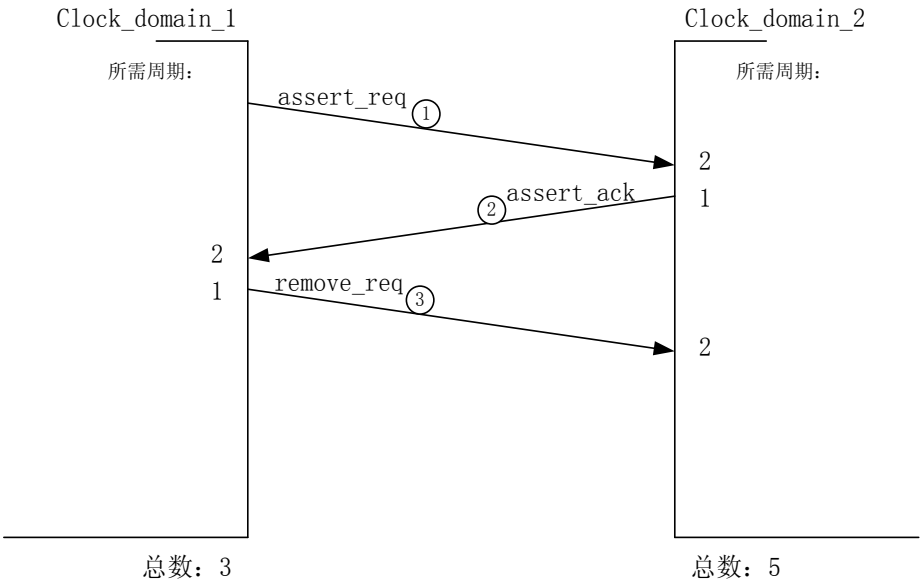


图 3.11 部分握手协议 I

在图 3.11 这种部分握手方法中，时钟域 2 并不关心时钟域 1 何时中止它的请求信号。但为了使得这种机制成立，时钟域 1 中止请求信号至少要有一个时钟周

期长，否则，时钟域 2 就不能区别前一个请求和新的请求。只有当时钟域 2 检测到请求信号时才会发出相应的一个脉冲。由此模型可以看出，在时钟域 1 中需要 3 个时钟周期；在时钟域 2 则需要 5 个时钟周期。下面是该部分握手协议的传输时序：

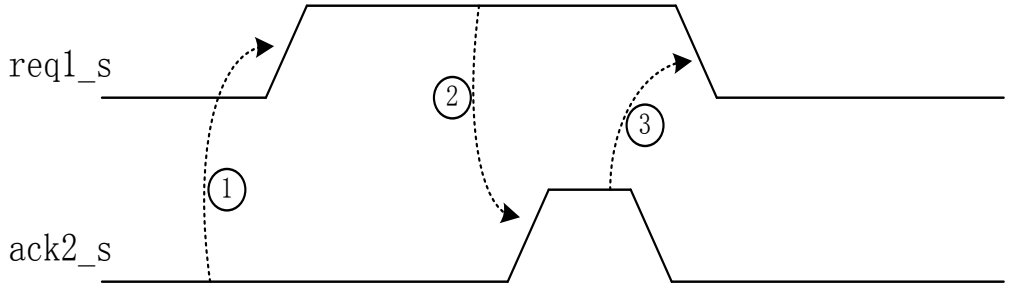


图 3.12 部分握手协议 I 的传输时序

可以看出，该部分握手协议时序省去了全握手协议中的最后一步，也就是 ack2_s 信号能够自动的把自己状态移除。而不是要等检测到 req1_s 的取消信号之后才能开始移除状态。

由上述的描述可以看出，部分握手协议 I 虽然在保证了同步传输的情况下，减少了握手所消耗的时间周期，但是不难看出这样的周期消耗还是不能够满足如今日益快速的 SOC 设计要求。这里引入另外一种部分握手协议 II，握手次序如下所示：

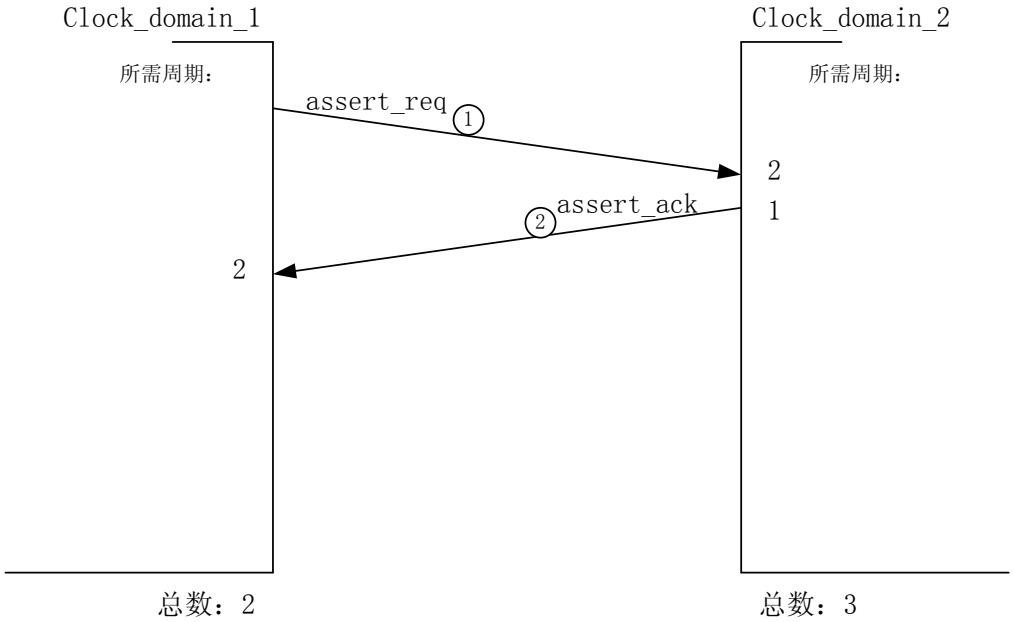


图 3.13 部分握手协议 II

该部分握手协议，可以减少一些时钟周期，因为此种形式的握手协议只存在两次的 req-ack 交互，时钟域 1 置位 req 信号，时钟域 2 要用 2 个时钟周期来进行

同步，同时消耗 1 个时钟周期置位 ack 信号，同理时钟域 1 也需要 2 个时钟周期来进行同步该 ack 信号。所以，传输一个数据需要源时钟域两个时钟周期加上目的时钟域的三个时钟周期。握手的时序如图 3.14：

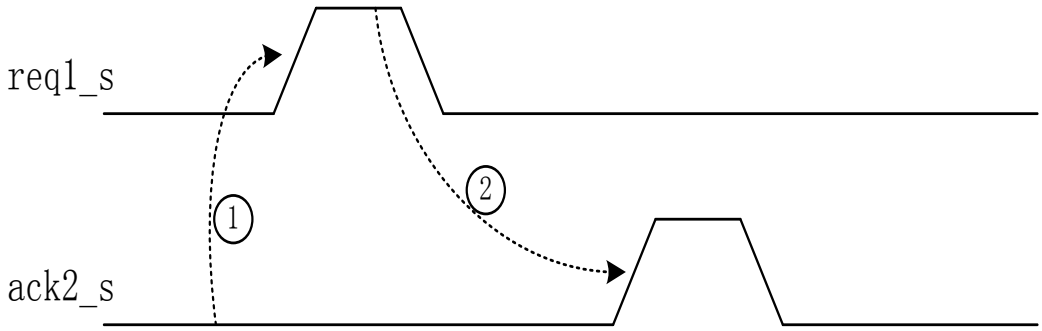


图 3.14 部分握手协议 II 时序

可以从该部分握手协议模型的时序图中看出来，req_ack 都是只交互一次，相比全握手协议，其省去了流程中的最后两部移除步骤，两个信号 req_ack 在 assert 一段时间之后，能够自动的移除掉本身的置位状态，而不存在相互检测取消的步骤。

以上总共介绍了三种常见的握手同步设计，通过上述的描述分析，可以基本总结出：对于全握手同步设计，其适应性和稳定性是最好的，但是传输单个数据的时间消耗却是最长的；经过优化的部分握手同步设计 I，其适应性和稳定性没有全握手那么稳定，原因是减少了时钟域之间的一些交互，但是这样却在传输一个数据的时候，相对减少了一定的周期消耗。传输的速率有所提高；最后一种优化的部分握手同步设计 II，其适应性和稳定性是最差的，但是传输单个数据所需的周期数却是最少的，因为其减少了一般的交互动作。所以设计者在进行 SOC 设计的时候，需要根据应用场合的要求，合理的选择握手同步设计进行 CDC 多 bit 数据传输。

3.2.2 异步 FIFO 同步设计

在许多情况下，多 bit 数据在 CDC 时需要“堆积”起来，因此如果使用单个保持寄存器无法完成工作。例如在某个时钟域的电路需要突发式的发送数据，接受电路则来不及进行采样。同理，如果突发式的接受一堆数据，而系统的发送端却不能满足，对于这类情况，为了能够最大限度的提高 SOC 中各个电路的效率问题，通常将会采用异步 FIFO 进行这种 CDC 的同步处理，让缓慢的数据先在一个存储单云堆积一定的程度，而这个时候，采样数据的电路则可以处理其他的事件或数据，等堆积到一定的程度之后，在由采样电路来进行一次快速的采样读取，

所以，对于如今的高速 SOC 设计来说，设计人员必然要考虑这种异步 FIFO 的同步设计方法。

下面来看看这样的一个典型的异步 FIFO 的电路结构[12][13]：

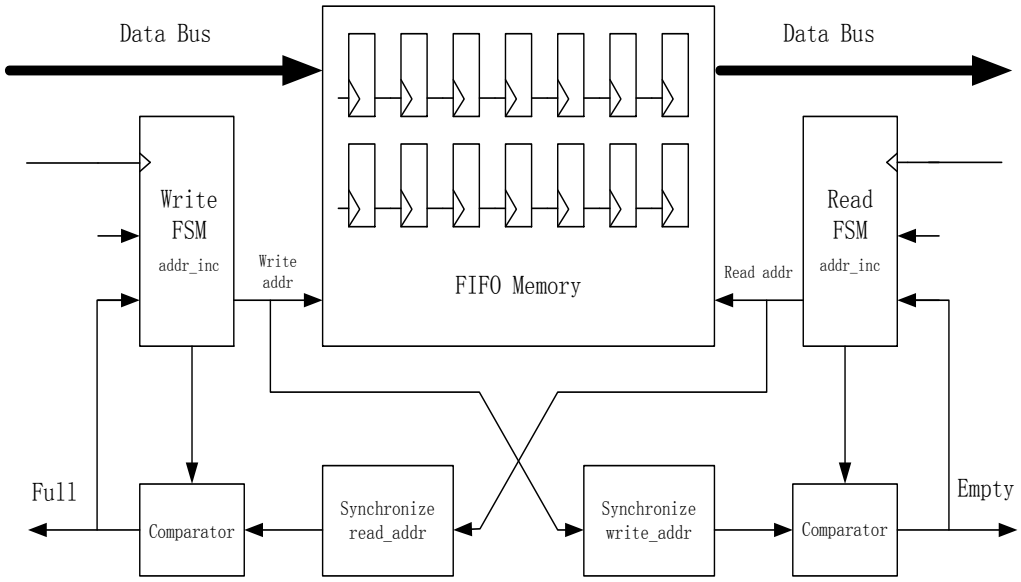


图 3.15 典型异步 FIFO 电路结构

如上一个典型的 FIFO 结构，其存储介质为一个双端口的存储器，性质为 RAM（只读存储）。可以同时进行读和写的操作。通过上述的典型电路结构，可以看到这样的一个异步 FIFO 能够很好的处理两个方面：速度匹配和数据宽度的匹配。这样既保证了 CDC 的亚稳态问题，同时也处理了位数宽度的不一致问题。

需要注意的是上面的 FIFO 结构中，如果采用一般地址线的二进制编码进行同步处理，则会碰到如前所述的输出数据不一致问题。因此在进行地址线等的传输时候通常需要对地址线进行格雷编码，然后再进行 CDC 的传输进入另外时钟域进行比较处理，从而在读写时钟域分别产生相应的空满标志位。这样就能够很方便的通过调节读写操作，避免 FIFO 的读空和写满等误操作。

3.3 异步复位信号的同步设计

对于数字设计电路来说，通常采用的复位方式主要有两种：同步复位和异步复位。对于同步复位，其复位和释放变化均与时钟信号的有效采样边沿保持一致，然而对于异步复位，复位不依赖于时钟信号，而释放却是需要时钟信号的采样完成。这样在复位信号在释放的时候，非常容易产生亚稳态的问题。

那么在此就来介绍一种通常的解决办法，如下图所示，复位信号同步设计。异步复位信号同步设计通常是为了防止由于在复位信号释放的时候出现的亚稳态问题。

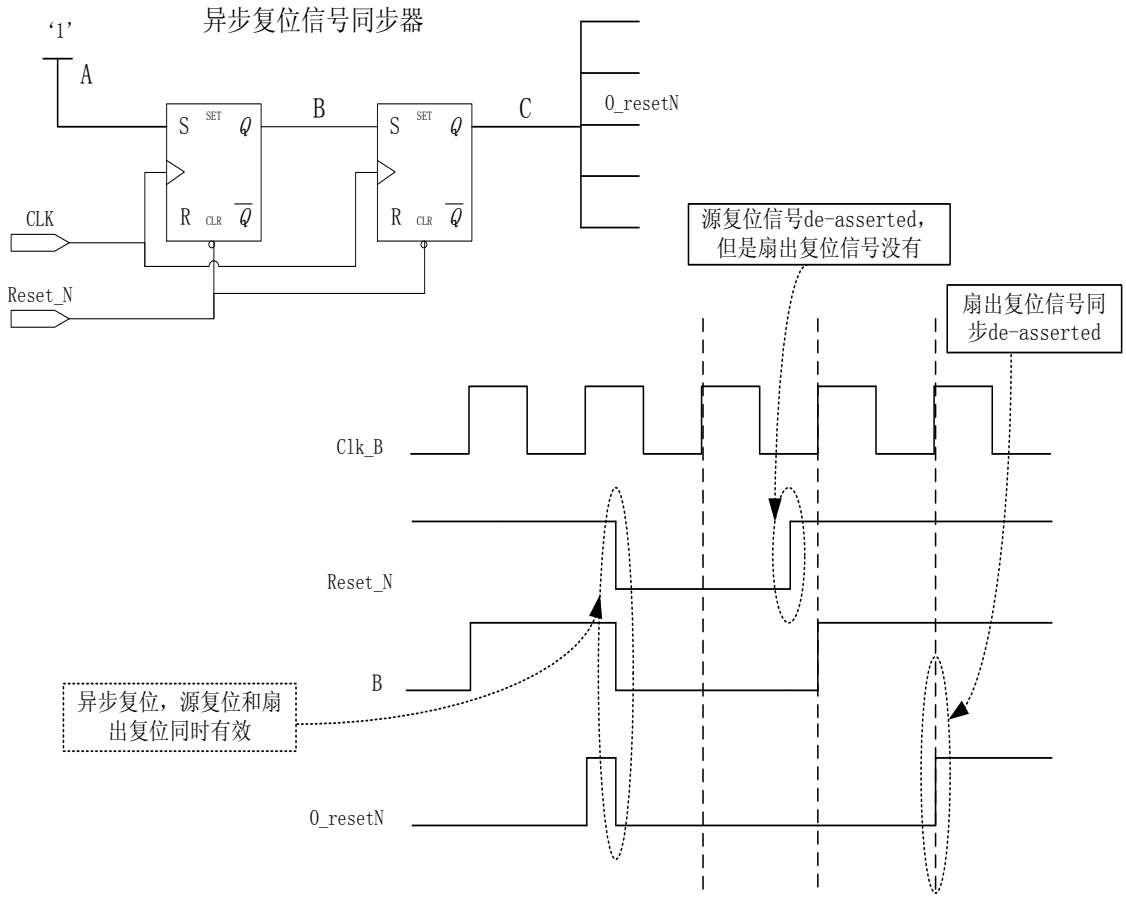


图 3.16 异步复位同步设计（同步释放）

尽管有这样的异步复位同步设计，但还是需要进行一系列的 CDC 验证，不仅需要进行相应的这些同步电路的验证，同时也应该进行一些其他的检查，比如有可能这样的异步复位信号在一个时钟域中进行了多次的同步等等。

3.4 CDC 同步设计的传统验证方法

对于以上所介绍的几种在数字 SOC 设计中经常能够碰到的 CDC 同步设计来说，了解了其基本的应用优势与限制条件之后，另一项重要的工作就是对他们的验证。在目前的数字设计流程中，能够进行这种专门的 CDC 验证的工具或者方法很少，一般都是通过传统的验证方法进行部分的辅助验证。设计前后端都要实施。下面是数字 SOC 设计的阶段性流程图以及几种传统的电路验证方法的分布情况：

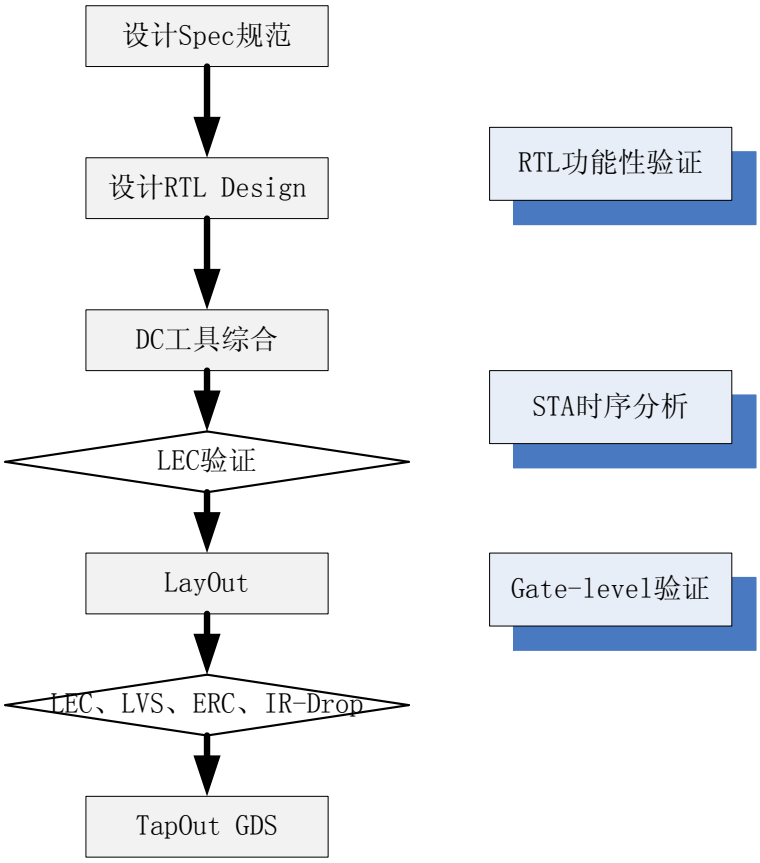


图 3.17 数字 IC 设计流程以及传统验证方法

对以上的数字流程中存在的传统验证方法，如果要单独的针对 CDC 同步设计进行检查分析的话，都存在一些明显的验证缺陷：

首先，前端的 RTL 功能性验证。这种验证方法在 RTL 阶段不仅需要读取相关的 RTL 设计文件，同时也需要搭建合适的 TB（test bench）验证环境。另外一个最大的限制条件则是：需要添加足够多的测试向量才能覆盖数字设计中尽可能多的 CDC 传输路径。对于一个大规模的 SOC 项目来说，这种创建大量专门测试向量的验证手段显然是不合适的。

其次，设计后端 STA 时序分析。主要的 CDC 验证弊端：一则是因为其处在芯片设计的后端，不能够对 RTL 级进行早期的验证，同时对于异步传输路径通常是不进行检查的。基于这两点可知，其显然也是不可取的。

最后，Gate-level 验证。在数字设计的后端，往往都需要通过搭建综合网表加上 sdf（standard delay format）延迟文件的形式的 TB 验证环境用于检验设计的时序。但是这种方法和前面的功能性验证都属于一种动态类型的验证方法。在专门 CDC 检查方面存在与其相同的弊端。同时还有重要一点：该种验证方法同样也处在数字设计流程的较后端，对于发现和解决 CDC 问题存在一定的限制。

综上所述，目前数字设计流程中，所采用的几种传统的验证方法在专门的 CDC 早期检查以及验证充分性方面都存在较大的不足。

3.5 本章小结

通过前面分析的亚稳态在不同 clock-domain 以及 power-domain 的恢复机理，并且结合实际项目工程中的具体情况，本章节中给出了几种不同应用场合下的同步设计类型。

对于例如控制、使能等类型的单 bit 数据信号 CDC 传输时，着重分析了该信号在快到慢以及慢到快的时钟域关联情况下，完成必要的同步通常会采取的设计要求以及电路模型。其中慢时钟域到快时钟域情况下的电平同步设计以及边沿同步设计，当目的时钟域频率与源时钟域小于两倍的情况下，单 bit 的异步输入信号必须保持至少两个源时钟域的周期，这样才能不会产生数据的丢失情况。另外从快到慢时钟域的情况下，由于通常采用的无反馈型的脉冲同步设计对输入的相邻两个脉冲间隔要求比较苛刻，所以设计并引入了另一种适应性较强的带 busy 反馈性脉冲同步设计，依据反馈的 busy 信号，脉冲数据产生端能够很清楚的知道什么时候可以发送下一个脉冲信号，从而消除了脉冲间隔问题。

而当 CDC 传输的数据为多 bit 的总线数据、地址总线等的情况下，重点分析了在目前 SOC 工程项目中常用的握手同步设计以及异步 FIFO 的同步设计。其中握手同步设计根据应用条件的不同，分为了全握手和部分握手两类。全握手适用于稳定性要求高、传输速度要求低的场合；而部分握手则是优化省略了全握手中的部分协议传输，使得其牺牲了一定的稳定性，来提升数据的传输速度。此类部分握手同步设计主要应用在高速的 SOC 设计中。当然，对于源、目的时钟域数据宽度不同以及时钟频率相差悬殊的情况下，特别是在基于 ARM 等核心处理器的 SOC 设计中，必然要采用常见的异步 FIFO 的同步形式来完成数据的 CDC 传输。

除了以上的数据信号的同步之外，在 SOC 的设计中还存在着异步复位信号的同步情况，如果不采取同步措施，由于其在复位时为异步、，而置位却要依赖时钟信号的采样，这样同样也会出现亚稳态的现象。本章同样给出了数字 SOC 项目工程中常见的异步复位信号的同步设计电路模型，使得该类复位信号能够异步复位、同步置位。

最后结合以上的几种同步设计类型，具体分析了在如今数字 SOC 项目中传统的几种验证方法对其的检验程度和覆盖程度。前端功能性验证以及后端 gate-level 验证都存在验证环境的复杂、CDC 验证的不充分以及验证时间节点的不合适；而后端的 STA 时序分析则不能充分覆盖 CDC 路径，并且验证时间节点也太晚。使得项目不能充分验证这类 CDC 传输路径以及同步设计。

该章节内容对数字 SOC 设计提供了一定的分析借鉴意义，同时为后续章节专门 CDC 验证方法的提出做出了必要的准备。

第四章 基于 SpyGlass 的数字芯片 CDC 静态验证

对于现在的数字 SOC 设计，验证工作[14][15]在整个设计的流程中扮演着越来越重要的角色，其目的是为了检验设计是否与设计规范一致，是否实现规范中包含的各种功能，从而尽早的发现设计中的问题，保证流片的成功几率。

基于上章节所讨论的 CDC 同步的问题以及传统的验证方法分析情况，目前还没一种比较成熟和良好的专门 CDC 验证方法在 SOC 设计早期流程中实施[16]。本章节中，重点介绍一种新型的静态 CDC 验证方法，不仅能够在设计的 RTL 早期阶段保证同步设计的正确实施和良好的代码质量，而且也能够在功能上保证这些同步设计能得到验证。

4.1 SpyGlass 的 CDC 静态验证方法分析

通过上面的 CDC 同步设计分析以及相应的传统验证方法在该方面的一些不足分析，同时为了设计的需要，必须提出一种新型的、RTL 级、穷尽 CDC 传输路径的验证方法，首先对于目前数字 SOC 复杂、多变的设计特点，这类专门的静态 CDC 验证方法必须能够在最大限度上进行分析。而且相比较传统的验证方法，其必须只能在依据 RTL 代码的情况下独立完成验证工作。

4.1.1 静态 CDC 验证方法需求特点

对于当今的设计和验证趋势，设计规模以及设计周期不断提出更加严苛的要求，此种情况下提出的 CDC 静态的验证分析手段，主要能够在设计的早期 RTL 设计的阶段就开始相关的 CDC 验证检查，及时的发现设计中的一些缺陷。对于芯片的设计人员来说，考量一种静态验证方法应该从以下几个主要的因素[17]去入手选择：

1. 能够充分的验证各种 CDC 问题。
2. 能够支持各种标准的同步设计，包括握手同步设计和 FIFO 同步设计等。
3. 灵活的适应设计人员对标准同步设计的一些特定配置参数。
4. 能够从功能上去验证握手和 FIFO 同步设计的完整性。
5. 良好的干扰管理技术，通常用于管理一些 SOC 中常见的 CDC 干扰问题。

当然，除了上述的这几个比较关键的要求之外，还有一些其他方面也是需要满足的：比如对于一个结构层次十分复杂的数字 SOC 电路来说，在发现这些 CDC

传输中的问题后，验证方法还要够很快捷方便的定位到具体的设计点中去等等。

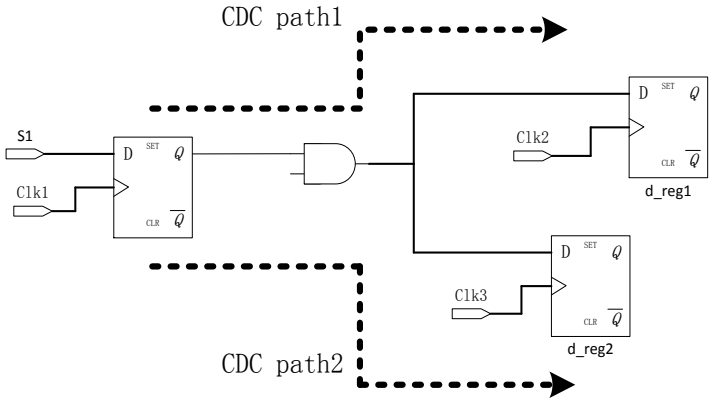
4.1.2 SpyGlass 的 CDC 静态验证方法分析

基于静态 CDC 验证方法需求分析，在这里引入基于 SpyGlass 的一种 CDC 静态的验证方法，并且简单的介绍与分析其相比较传统的验证方法，在专门 CDC 同步设计验证上面所具备的一些优势。

相对于数字设计流程中传统的功能性验证以及后端的网表级验证手段，主要不足是所能覆盖的 CDC 传输路径是有限的。而 SpyGlass 的 CDC 静态验证方法则具备了以下几种优势：

第一， 由于是一种静态的验证方法，所以相比较功能性验证等方法，就不需要搭建 CDC 验证所需要的测试环境，而只要 RTL 设计完成之后，就能够开展相应的专门 CDC 静态验证工作。

第二， 在 CDC 传输路径的覆盖方面，传统的功能性验证方法等可能需要创建非常多的验证测试向量来进行满足，而基于 SpyGlass 的 CDC 静态验证方法，对于验证模块中所有的包含时钟信号的触发器等时序元件，都会进行相应 CDC 检查分析。如下图所示：



传统功能性验证方法：
cdc_path1 对应一组测试向量a；
cdc_path2 对应一组测试向量b；
...

SpyGlass的CDC静态验证方法：
能够穷举cdc_path1以及cdc_path2等**所有的cdc传输路径**。

图 4.1 SpyGlass 的 CDC 静态验证方法与传统验证比较

此外，对于设计后端的 STA 时序分析验证方法以及网表级的验证方法比较而

言，SpyGlass 的优势就更加的突出：

第一， 由于处在设计流程的后端，STA 时序分析和网表级验证一旦发现 CDC 问题，相对来讲比较被动，对设计周期和流片都会有影响；而基于 SpyGlassc 的 CDC 验证则处在设计的最前段 RTL 级。发现问题能够尽早的修正。

第二， STA 静态时序分析这种方法虽然不需要搭建相应的验证环境，但是其检验的重点则是同步传输，而对于 CDC 路径不能够进行验证覆盖，但是 SpyGlass 的验证方法则重点在于 CDC 之间的传输路径的验证。

基于上面的一系列分析得出：如果在目前的设计流程中实施这种基于 SpyGlass 的 CDC 静态验证方法，对于数字 SOC 工程的 RTL 设计质量以及设计周期来说是一个很好的帮助。

4.2 SpyGlass 的 CDC 静态验证环境的设计

经过上述的 SpyGlass 的静态验证方法的一些简单分析，应该能够很清楚地理解这种专门的 CDC 的验证方法在目前的设计中存在的巨大潜力，从设计工程的角度出发，由于其对于相关的验证设计的输入以及设计要求相对简单，通过自身的内部构建模型化，就能够实施相应的 CDC 验证；并且能够提供对于大规模数字 SOC 工程的解决方案。

4.2.1 SpyGlass 的数字芯片 Top-Level CDC 验证环境的搭建

首先要知道这种基于 SpyGlass-CDC 的验证方法是一种静态的检查方法，既然是静态的检查验证手段，不存在相应的验证平台需要，所以，只需要读进去相关的设计要求以及相关库文件。因为对于 SpyGlass-CDC 来说，即便是没有任何工艺库文件制定的设计，SpyGlass-CDC 也提供了一套属于自己的库文件用来进行综合设计使用。对于一个具体的模块或者 IP 来说，可以读入所有的设计文件以及例化的库文件，这样对于编译设计来说，能够保证完整性。

对于 SpyGlass-CDC 验证环境来说，首先要完成第一步准备的工作，Setup 初始化步骤。通过这一步 SpyGlass 需要读入相应的设计，然后在去定义设计中所有使用的时钟信号和复位信号。同时能解决相关的 black-box（黑盒型）问题。

首先读入相关的 RTL 设计，对于大规模的数字芯片，通常在进行 RTL 设计读入的时候需要使用如下所示方法：

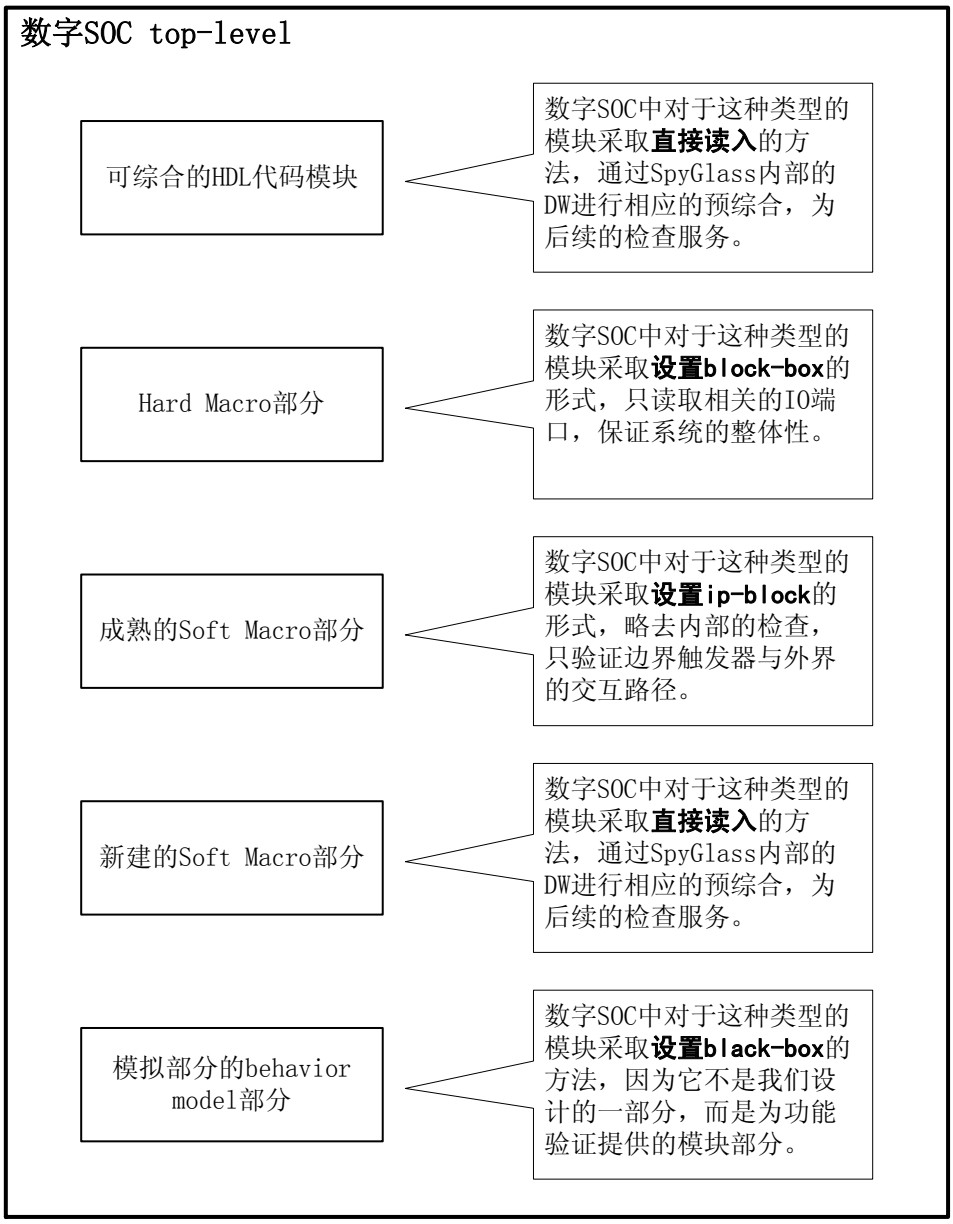


图 4.2 数字 SOC 内部各种模块的读取处理方式

对于设计读入的过程中，涉及到几个基本的读入控制、设置命令：

- set_option dw yes

  该命令是选择是否使用 SpyGlass 内嵌的 design ware 用来进行预综合。

- set_option stop <模块名>

  该命令是用来选择是否对一些设计模块进行 black box（黑盒）处理。

- ip_block -name <g3_shell>

  该命令使得 SpyGlass 虽然读取所有该模块的设计文件，但是如果已经对其进行了模块级的 SpyGlass 验证的情况下，就可以只检测其边界的触发器与外部模块之间的交互问题。

经过上述的过程，SpyGlass 完成了所有的设计文件读入步骤，经过

SpyGlass-CDC 自身的预综合，可以得到了一个基本的设计的编译结果以及网表结果。这就是 RTL 设计的一个数据堆。为后面的静态检查做准备。

在 Setup 这一步骤中，除了对这些不同设计文件的读入，还需要做所有设计中的时钟信号和复位信号的定义文件准备。只有在有了约束文件，才能让 SpyGlass-CDC 工具得知具体设计中的时钟域分类和同步复位以及异步复位信号，然后进行相应的静态分析。数字芯片的时钟信号定义架构如下。

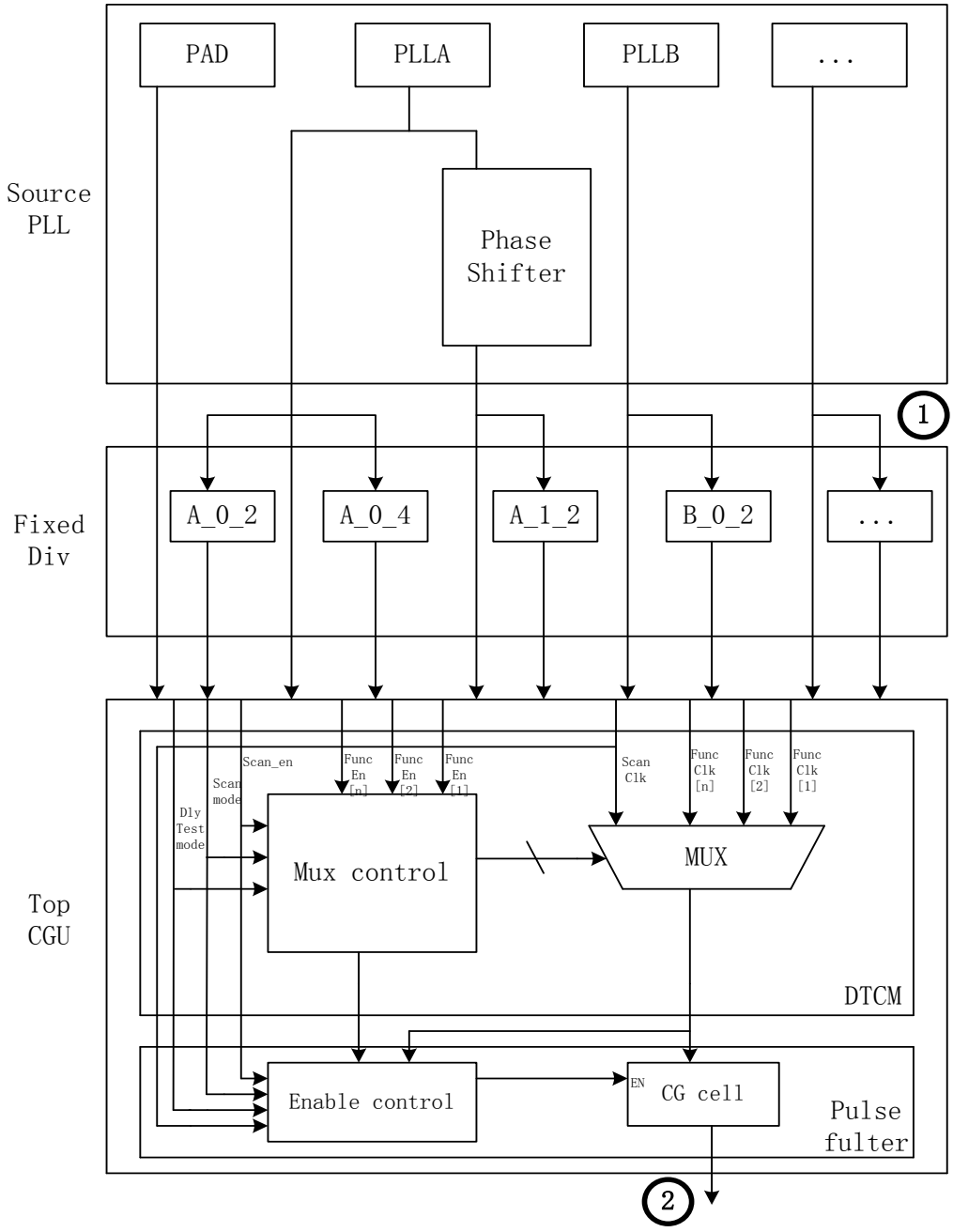


图 4.3 数字芯片 top-level 时钟信号的结构定义

上面的时钟的结构，是在数字芯片设计工程的顶层常见的一种时钟的架构，对于 CDC 验证环境的构建来说，需要定义在图中 1 处所有的 PLL 的基本时钟信

号，另外还有 2 处的 Top CGU（clock generate unit）从 DTCM 模块中输出的信号，因为这两组信号都有可能直接送向 SOC 中的触发器的时钟信号段，或者经过一些组合逻辑或时序逻辑在送给触发器，通过这样的一个定义，大体上能够确定从 SOC 的顶层出发，所有的时钟域都有哪些。

时钟信号的定义方法：

clock –name top.clk –period 10 –edge {0　5}

上面的描述就是定义一个基本的时钟信号的 SpyGlass-CDC 语法描述。其中 -name 后面是时钟信号的全路径，其中 top 为这个 SOC 的顶层模块名字，-period 表示时钟的频率为多少，默认的频率单位为“ns”，-edge 表示时钟信号的上升沿和下降沿时间比例，也就是所说的占空比。

复位信号的定义方法：

reset –async -name "top.RESET_ALL_N" -value 0

上面的这个描述是定义复位信号时候经常用的方法。其中-async 表示该信号是一个异步复位信号，如果是同步复位信号，则应使用-sync 选项，-name 后面跟的是复位信号的全路径名字，-value 表示该复位信号的值，0 有效或者 1 有效。

以上的 SpyGlass CDC 验证环境搭建的第一步 setup，读取有关 RTL 设计文件，并且完成时钟信号和复位信号等约束文件准备工作，在这里，我们的前提是已经知道了设计中存在的所有时钟结构和时钟信号、复位结构和复位信号的定义，直接去书写这些约束文件，但是如果验证人员不是该模块的设计者，或者对于基带 SOC 设计的顶层的时钟和复位的层次结构不是非常的了解，就不能用上述的这种“正向”的办法去完成 setup 步骤。

为了解决上述的这个问题，SpyGlass 提供了一种特别的方法，能够在读取了所有相关的设计顶层设计文件之后，它提供一个 CDC_setup 的规则，其允许工具自动的去抽取所有设计文件中出现的可能的时钟信号和复位信号，同时生成两个对应的临时文件（autoclocks.sgdc 和 autoresets.sgdc），其中包含了 SpyGlass 自认为是相关触发器时钟和复位信号的综合，不过这些要经过设计者的复查才能真正的使用这样的临时约束文件，推荐使用正向的方法去收集设计顶层时钟和复位信号。

SpyGlass 中关于设计文件以及约束条件的读入步骤，其 GUI（用户交互）界面如图 4.4 所示：

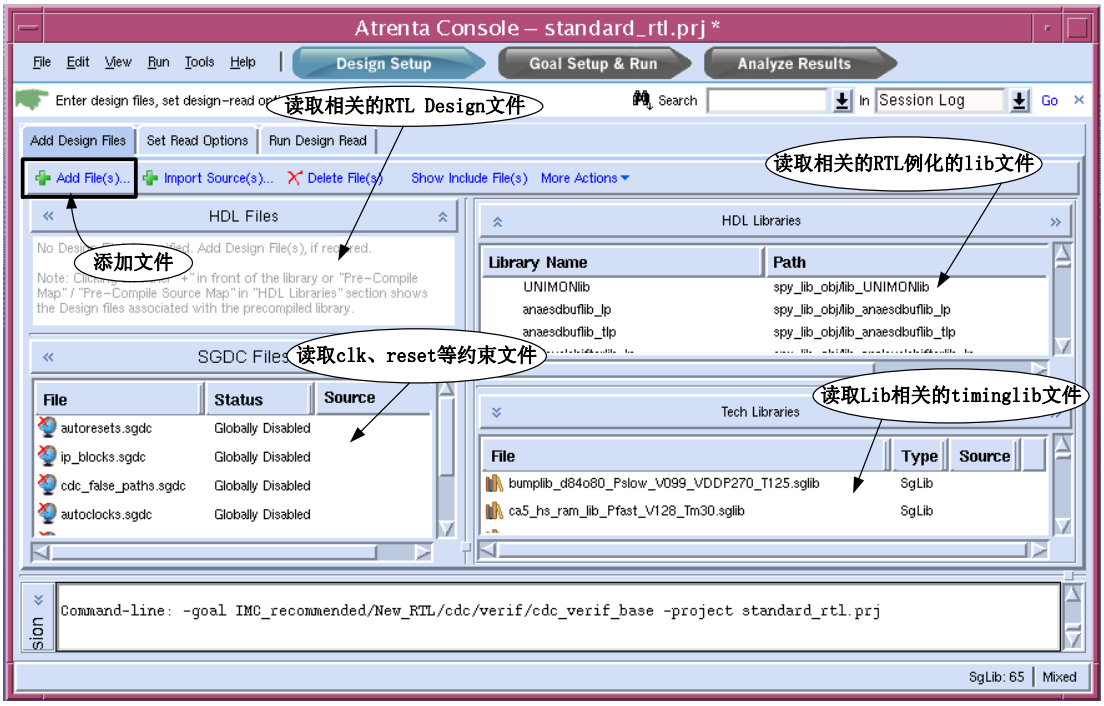


图 4.4 读入 RTL 文件和约束文件的 GUI 界面介绍

下面的是 SpyGlass 所提供的设置读取设计文件时候的控制选项，GUI 如下：

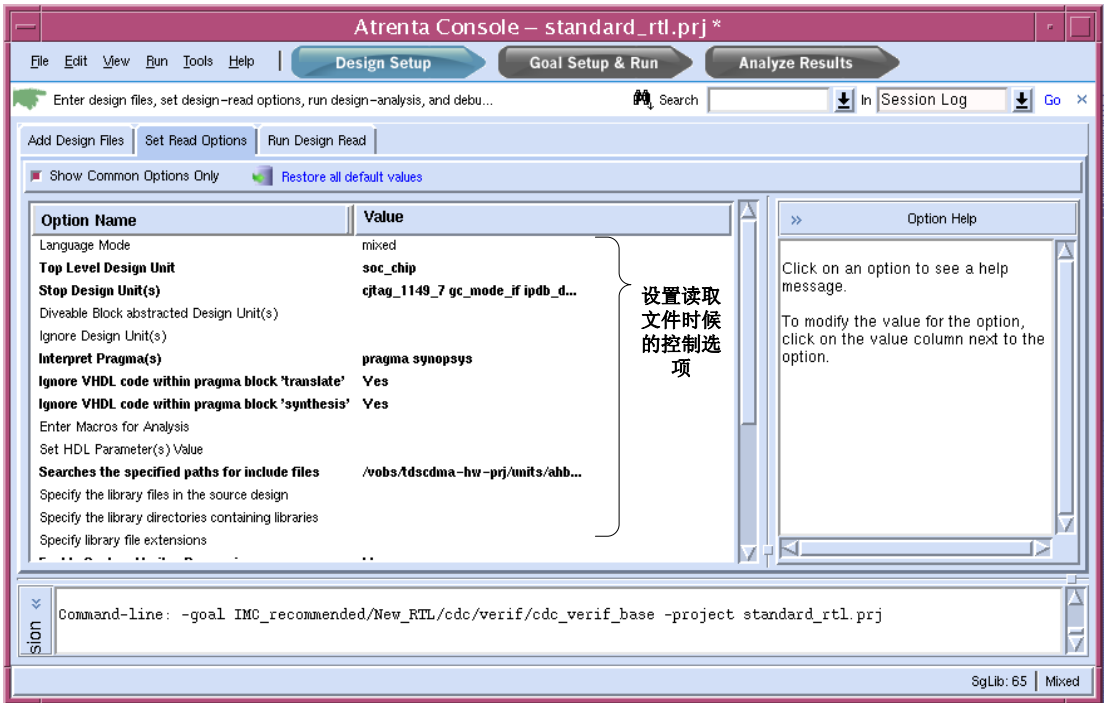


图 4.5 读入 RTL design 的时候控制选项的设置界面介绍

最后是 setup 这一步骤的最后一个阶段，那就是进行 RTL 文件以及相关 sgdc 约束文件和 sglib 的读取形成数据堆的步骤。SpyGlass 所提供的 GUI 界面如下所示：

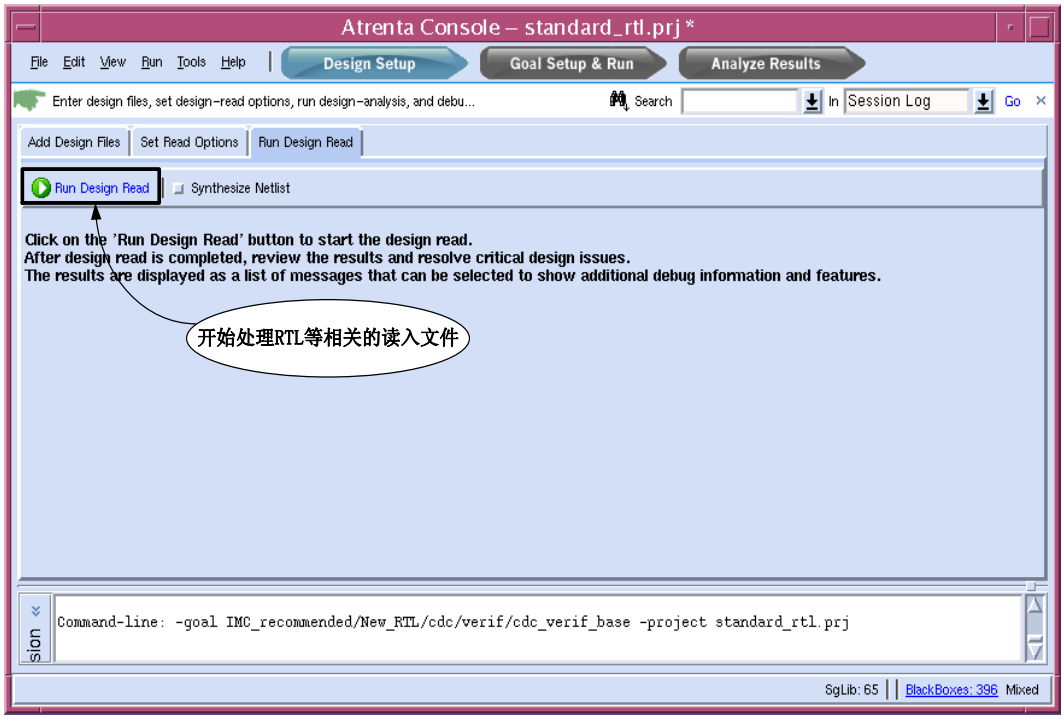


图 4.6 处理相关的 RTL 等文件的 GUI 界面介绍

经过上述的设计文件读入以及时钟和复位定义文件的准备，下面就是 SpyGlass 的 CDC 验证环境的约束检查步骤，通过这一步，就可以检查所提供的设计顶层时钟和复位信号的约束文件，是否存在相关的问题，该步骤 GUI 如下：

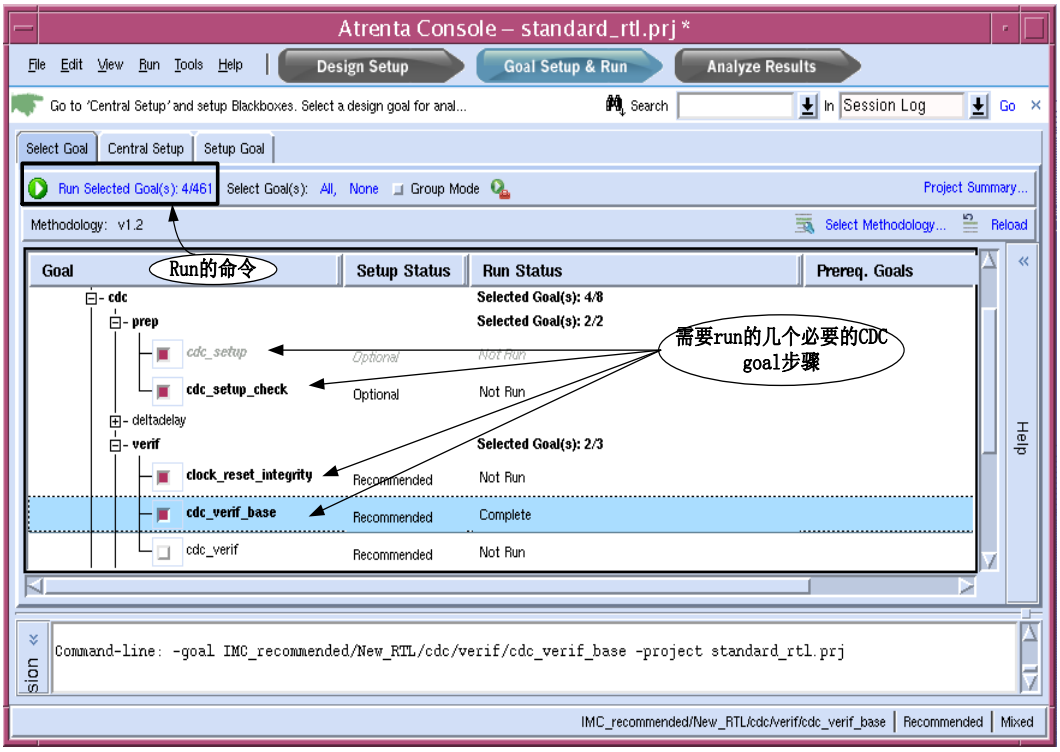


图 4.7 时钟和复位信号的检查 GUI 界面介绍

通常在这里我们会检验出我们的时钟等约束文件所存在的一系列问题，或者是实际 RTL 设计中的一些问题。如下所示：

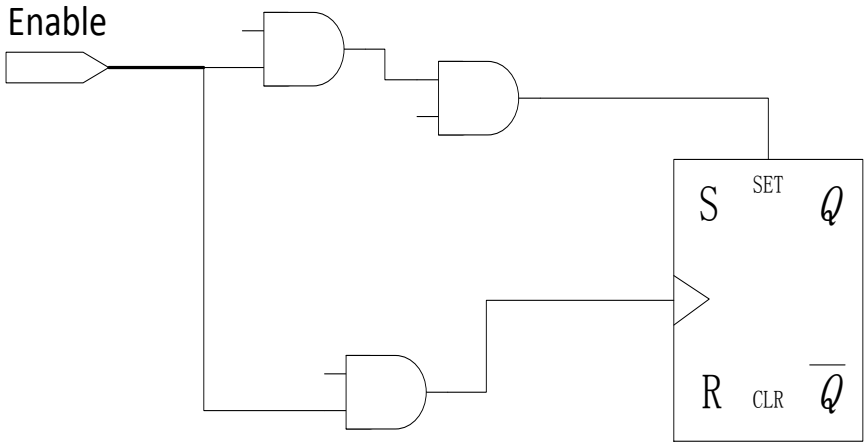


图 4.8 时钟约束等检查问题 1

碰到上述的问题，则需要重新定义我们约束文件，使其打断其中的任何一条路径，否则对于后续的 CDC 静态验证会有干扰。

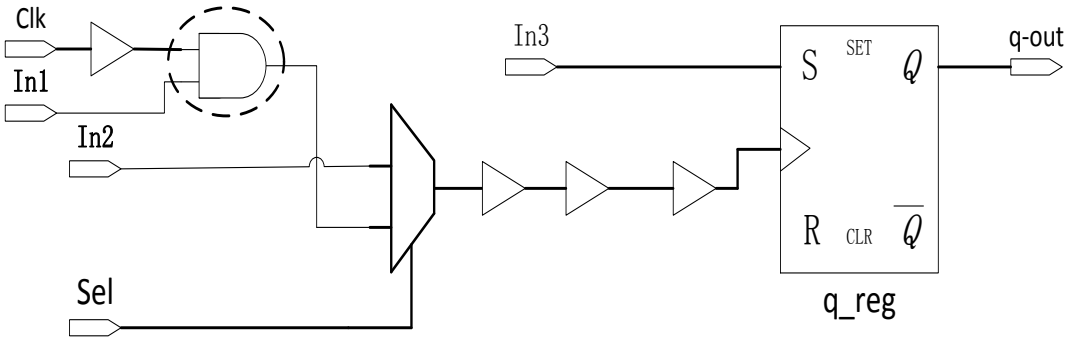


图 4.9 时钟约束等检查问题 2

此种问题主要是由于在时钟树的路径上出现了一些逻辑门，通常这些会导致时序问题或者产生一些毛刺，所以在这里我们需要通过修改设计拿掉该门或者修改我们的约束使其允许这个逻辑门在时钟树上的出现。

最后的步骤，就是进行 CDC 传输的检查，SpyGlass 会穷举之前预综合的网表中所有的触发器或者锁存器 CDC 传输的路径并且实施验证。通过与其自身内置的 CDC 同步设计电路的结构进行静态比较，同时也会加入在约束文件中的控制语句，通过这个步骤，产生一系列不满足的 CDC 同步设计电路结构的验证结果，其 GUI 如下：

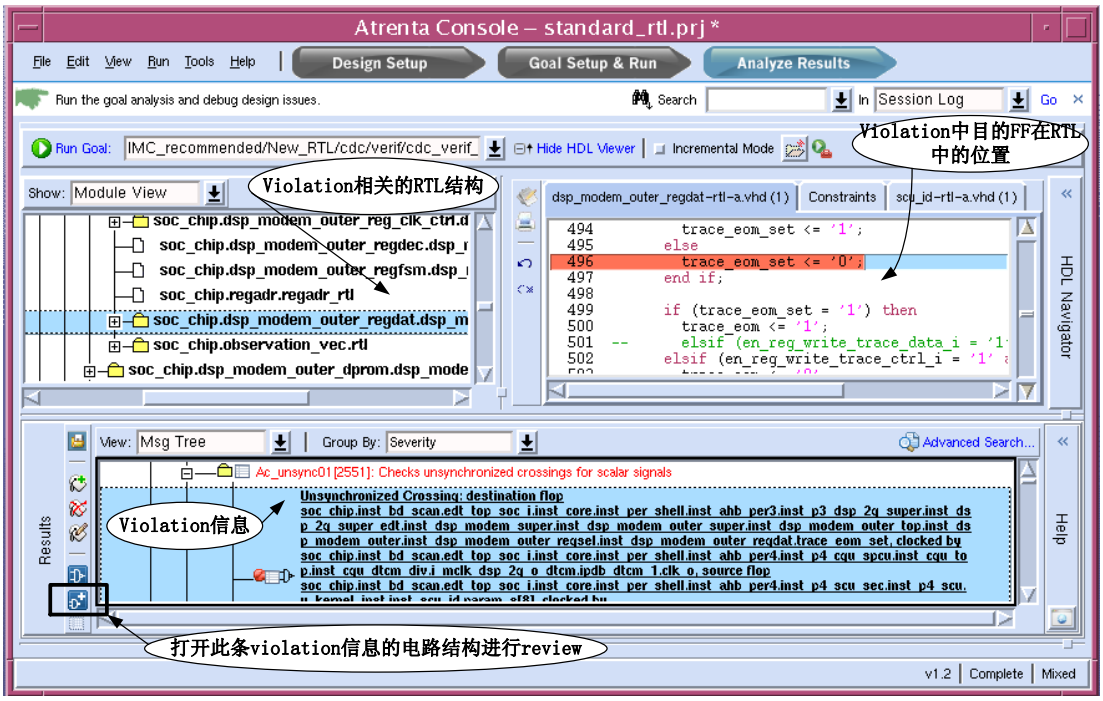


图 4.10 CDC 检查结果的 violation 信息 debug 界面 1 介绍

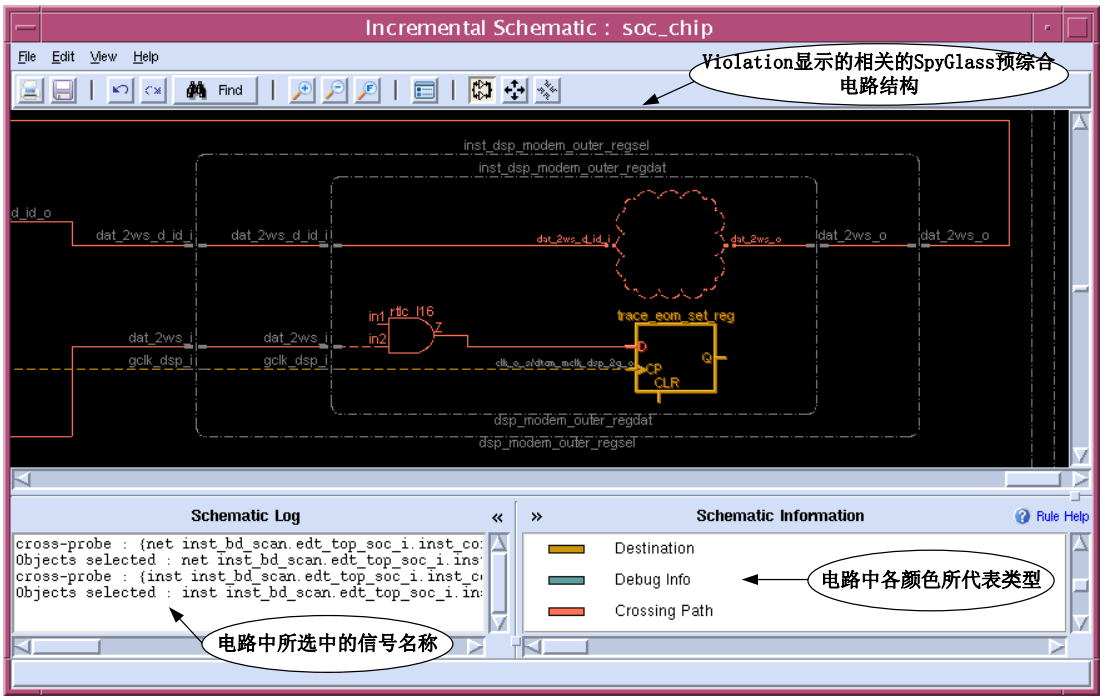


图 4.11 CDC 检查结果的 violation 信息 debug 界面 2 介绍

完成了上述步骤的 CDC 验证之后，就要开始分析验证结果，从而来修正具体的 RTL 设计。这里 SpyGlass 所提供的有两种能够方便解决问题的降噪机制。一种是排除（waiver）的形式；一种则是设置错误路径（false path）的形式。

对于单条目的信息，通过设计者的复查和设计目的对照，如果确定是设计规范在设计中能够容忍的时钟问题，那么，就可以采取第一种做法，使用 waiver 来去屏蔽掉该条信息，这样做的机制是：通过添加这样的 waiver 控制语句进入到的

约束文件，当开始进行下一次的再次执行的时候，SpyGlass 其实仍然会继续进行这条信息所对应的电路 CDC 的检查，并且会出现在验证结果的初始列表上面，但是由于有了 waiver 降噪控制的存在，SpyGlass 就会开始过滤，重新产生一个最终的问题列表文件，这个最终文件上就过滤掉了约束文件中所添加的 waiver 项，使得验证者在对最终的问题列表进行复查和设计对照时，就看不到该 waiver 屏蔽的问题结果条目，这样，就大大增加了 CDC 验证和解决问题的效率。Waiver 使用如下所示：

Waiver –rule “<rule-name>” –msg “<result message info>” –comment “<add why you set such waiver>”

当然，对于 waiver 这样的降噪处理手段，其优点是在于：能够在排除该条信息对应的电路结构的情况下，同时不会对相关的部分设计电路的 CDC 检查造成不必要的影响，如果去使用这样的处理方法得到的验证结果，必定是最安全和最干净的 CDC 验证结果。但是对于如今大规模的数字芯片而言，CDC 验证的初始结果必定也是一个巨大的问题列表，如果去这样逐条的进行 waiver，对于验证和解决问题效率而言也是十分低的。

SpyGlass 的 cdc_false_path 命令，通常使用这种手段的优点有以下几点：

首先，cdc_false_path 提供了三种控制选项，-from、-through、-to，使得验证者能够利用这三种选项去定位一个十分具体的设计电路 violation 信息，以此，其大大增强了屏蔽或者过滤 violation 信息的灵活性。然后，该控制语句是一种能够被检查的规则所理解的。如上述的 waiver 降噪方法，其只是在运行结束之后，问题列表经过了再次的后期处理，因此 waiver 是一种不被 CDC 验证规则所理解的约束手段。

正是因为这种大刀阔斧式的降噪处理方法，使得在解决问题的时候避免了许多干扰信息，同时，由于其作用的范围是在规则在进行 CDC 检查的过程中进行的，其所定位到的传输路径或者节点上面的所有 CDC 验证行为竟会被全部取消，包括以这条路径或者节点作为中间节点的传输路径也被取消了验证行为。

总体来讲，当在进行一个大规模的 SOC 的 SpyGlass-CDC 验证时候，应该采取两者结合的做法进行降噪处理，这样不仅验证解决问题的效率得到了提高，同时 CDC 的验证质量也能得到相应的保证。

4.2.2 SpyGlass 相关的同步电路识别分类

经过上面的验证环境和步骤的介绍，现在来具体的看一下，在进行静态的 CDC 验证检查的时候，SpyGlass 通常都会对 SOC 设计中的哪些同步设计结构进行识别。

第一种，SpyGlass 进行静态的同步结构分析的时候，经常会首先进行检查的同步结构就是传统的触发器串联形式的同步设计。这其中包含了我们在第三章讲述的如电平、边沿、脉冲等基于这一同步原理的所有同步设计类型。

第二种，通常在 RTL 阶段，仅仅使用一个触发器来进行代表，而在后端综合的时候将会对这一类的触发器进行替换，形成同步设计的单元，也就是设计者自定义的一些同步单元。

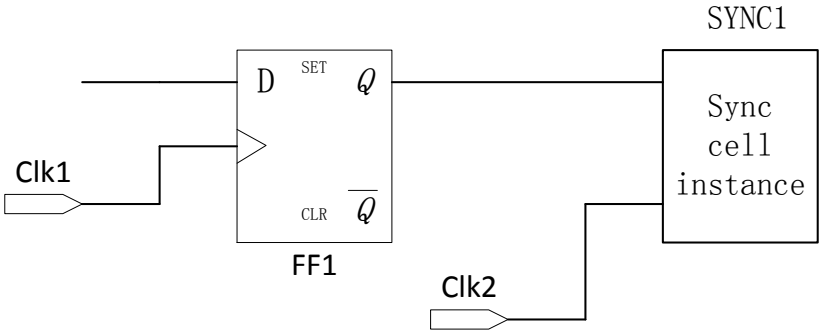


图 4.12 用户自定义的同步设计模型

对于这样的自定义同步设计单元，在 CDC 验证的是，可以通过 SpyGlass 提供的命令：sync_cell，来进行定义，上图就可以使用如下的表示：

sync_cell –name SYNC1

如果对于上述的电路模型不进行 sync_cell 的约束的话，CDC 检查将会显示该条路径上面 CDC 传输存在不同步的问题。因此，对于这些用户自定义的同步设计单元，我们应该在设计的早期就进行收集和约束。

第三种，就是在设计中也经常遇到的基于 qualifier 形式的同步设计，如图 4.13 所示：

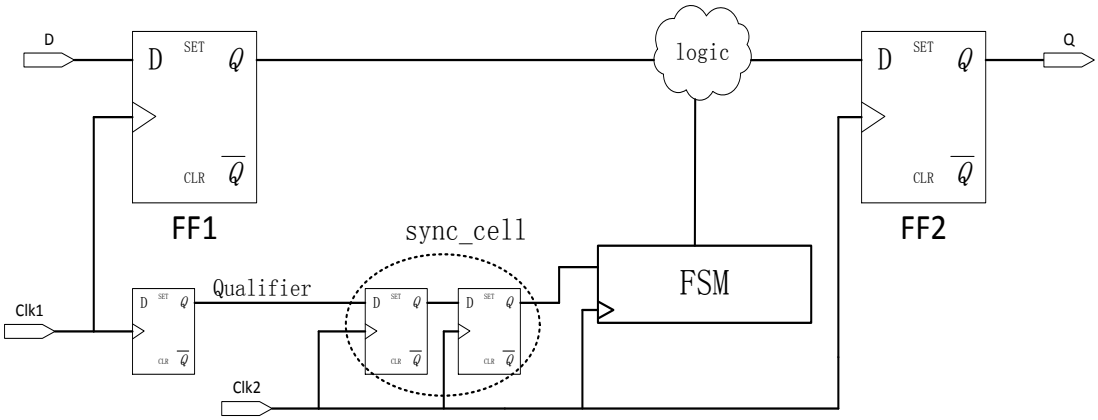


图 4.13 基于 qualifier 的同步设计模型 a

另外，还有以下的这种情况：

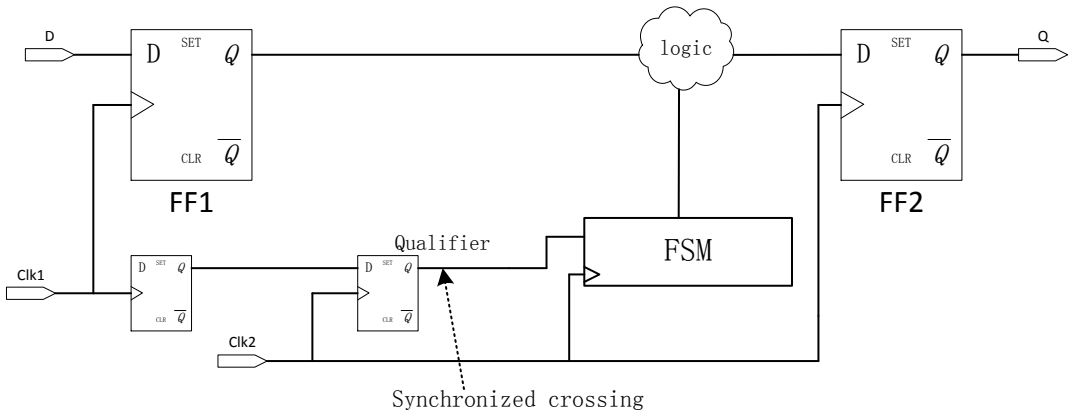


图 4.14 基于 qualifier 的同步设计模型 b

对于上述的基于 qualifier 的同步设计模型 b，可以使用 SpyGlass 所提供的 qualifier 约束手段，如果上述的 b 模型没有使用 qualifier 的约束，则会报出 CDC 不同步的 violation。当然如果确定该处是一个 qualifier 信号，就可以使用如下的约束来表示：

qualifier –name qualifier –from_clk clk1 –to_clk clk2

第四种，就是前面章节叙述的握手协议同步设计和异步 FIFO 类型的同步设计，在 SpyGlass 进行验证的时候，只需要设置两个参数就能够进行此类同步设计的识别，如下所示：

set_parameter enable_fifo soft

set_parameter enable_handshake yes

第五种，触发器控制端同步的设计类型，对于这样的设计在 SOC 中模块级是比较常见的。

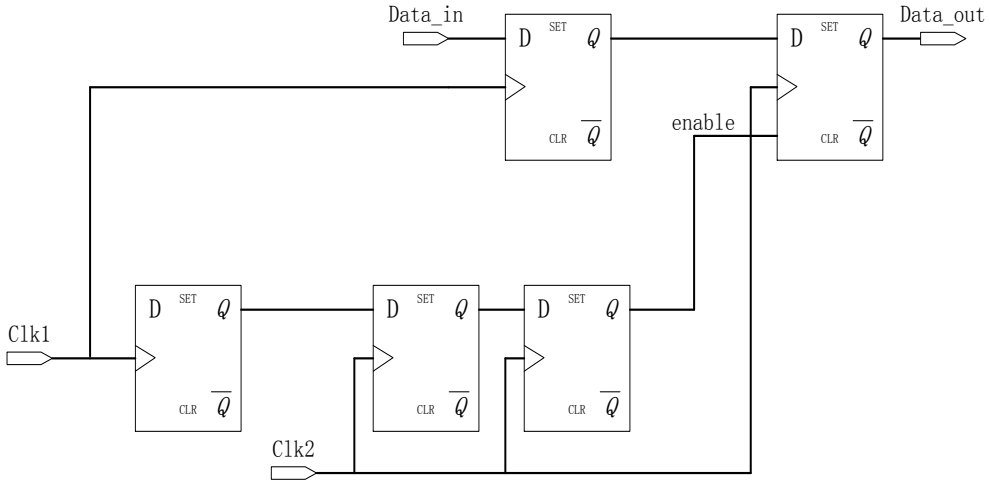


图 4.15 使能端同步的同步电路模型 a

同样由上述的这个同步设计思想，类似的处理方式还有许多，下面是在 CDC 静态检查中经常遇见的几种情况：

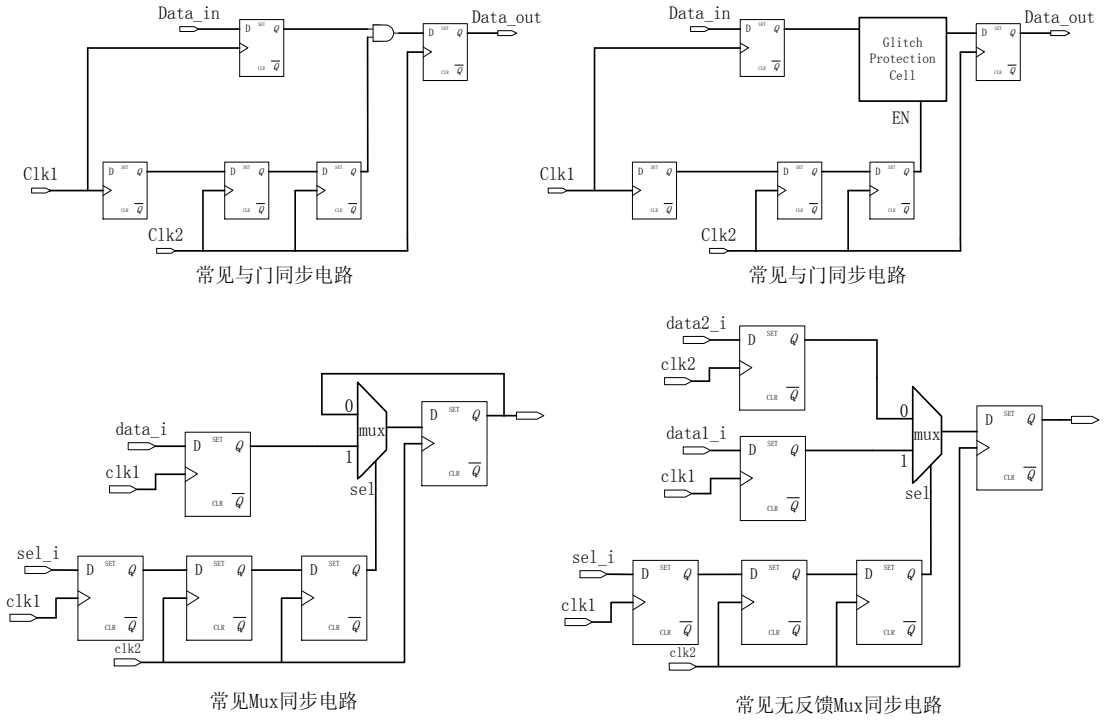


图 4.16 常见的触发器控制端同步设计模型 c

最后一种长延迟信号同步电路，之所以能够称之为长延迟的信号，主要是从该信号所处的时钟频率太低出发的，这里作这样的一个假设：一个 32KHz 的时钟域传输数据信号到 26MHz 时钟域时候，由于其 32KHz 的一个周期中会有 812.5 个 26MHz 的采样周期，因此，这样的 CDC 传输情况通常默认是十分安全的。在基带 SOC 设计的过程中，由于 32KHz 时钟信号是芯片内部的最低的基准时钟频率，所以对于 32KHz 时钟域的数据信号来说，不管进入哪个时钟域的时候，一般都不需要进行同步操作。当然，这种例子只是 quasi_static 信号的一个方面，属于 quasi_static 类型的还有几种情况，例如一个信号的值在绝大多数情况下是保持不变的；或者是 CDC 路径上的的逻辑块的值对于亚稳态不是很敏感等等。如下所示：

quasi_static -name <sig-name-list>

但是在使用约束 quasi_static 的时候应该注意到以下的几点，首先这种静态信号不能够穿越触发器、传统锁存器和黑盒模块。其次是使用这种约束的时候必须要十分的小心，因为其有可能会掩盖掉一些真正的设计缺陷。

如果一个源时钟域的数据信号没有经过上述的相关同步设计处理，在执行 SpyGlass 的 CDC 验证的时候，就认为是不同步的违例传输路径。

4.3 基于 SpyGlass 静态验证结果的分析与 RTL 修正

了解了上述的 SpyGlass 常用的 CDC 静态分析同步设计模型，那么就可以通过具体分析实际 SOC 中的电路结构，从而对其进行相关的同步设计模型匹配，如果能够找到相关的同步设计的电路模型，那么就可以从 SpyGlass 静态分析的角度来说它是一个同步的 CDC 传输路径，如果在电路结构上都不能够满足所述的同步模型匹配，那么其很有可能就是 SOC 设计上的一些缺陷。下面是基于 SpyGlass 的 CDC 验证结果，在设计中经常能够碰到的一些问题，以及分析和推荐的 RTL 修正方案。

4.3.1 qualifier 不存在的 violation 情况

第一种 SpyGlass 验证结果 violation 信息具体如下所示：

qualifier not found.

对于这种信息，是因为 SpyGlass 不能够在 SOC 的设计中找到这条 CDC 路径上相关的任何同步设计，如下所示的电路模型：

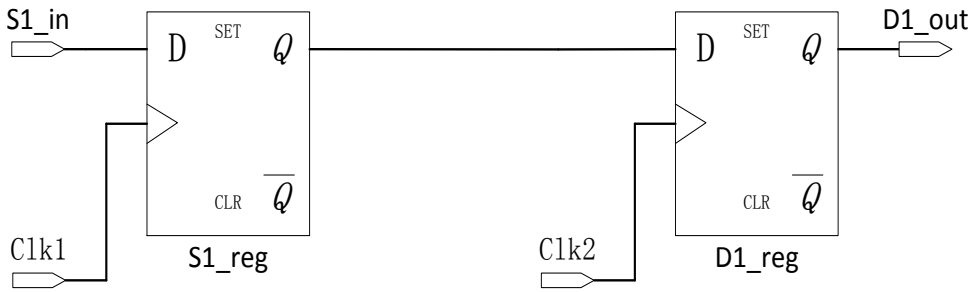


图 4.17 qualifier not found 电路模型

通常如果在 SpyGlass 的验证结果中出现这种 qualifier not found 的违例信息，从解决问题的角度来分析，首先要判断该条 CDC 传输路径上面是不是存在 quasi_static 约束；另外，排除了上述的这种情况之后，就应该考虑是不是源触发器和目的触发器的时钟信号是不是连接错误。

基于上述的分析，基本上可以得出此类 CDC 路径的 RTL 上面的修正方案：如果是 quasi_static 类型的信号，那么就只需要修正有关约束文件；如果是时钟的连接与设计规范所定义的有出入，那么就应该将此处的 RTL 时钟信号重新改正；当然如果都排除了上述的两种情况，那很有可能就是数字 RTL 设计中的缺陷了，那么设计者就应该在 RTL 中根据传输路径上数据的类型需求，插入相应的同步设计。

4.3.2 multi-sources 关联的 violation 情况

第二种常见的 violation 信息具体如下所示：

sources from different domains converge before being synchronized.

这种违例是最常见的。在系统的RTL设计阶段，经常会有信号相互穿通和叠加然后进行基本的逻辑组合操作，最后传送给另外的一些目的时钟域的触发器。

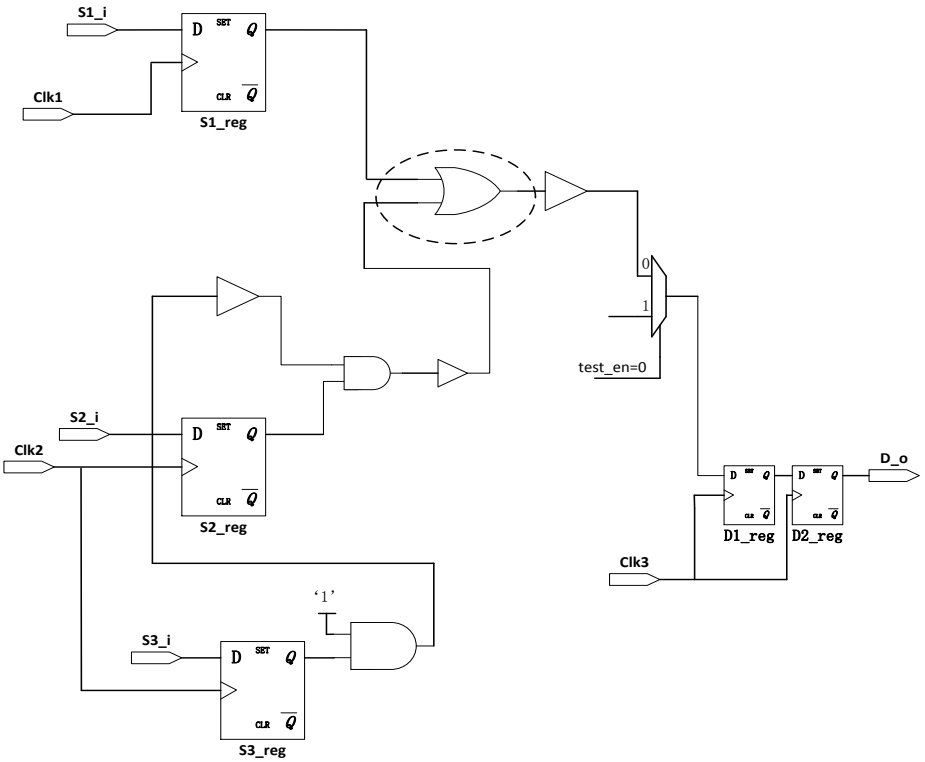


图 4.18 multi-sources 电路结构模型

对于这样的违例信息进行分析，首先应该确定几个源时钟域的时钟信号以及目的时钟域的时钟信号是否存在连接错误的情况，如上，如果S1_reg使用的时钟信号和目的时钟域一致的话，就能够先滤掉该条源路径继续进行分析，另外还应该注意这几个源之间的联系，例如S2_reg和S3_reg是处在同一个时钟域中，然后经过了一个mux产生输出，对于这样的输出也应该考虑是否需要添加一个触发器，保证在设计的顶层能够得到一个非组合逻辑输出的信号。

通过上述的分析，就可以考虑对于此类违例路径结构的基本的RTL修正方案：第一，要通过复查设计的设计规范确定时钟连接的正确性，否则需要进行RTL的修改，其次对于这样的电路结构，设计者可以在进行顶层模块或者IP之间的集成的时候，使其形成一个寄存器类型的输出，然后再送给目的模块或者触发器。基于上述的虚线内的逻辑，此种修正方案如下所示：

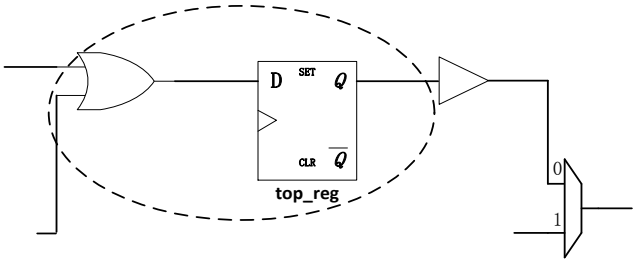


图 4.19 multi-sources 电路结构模型修正方案 1

当然上述的这种修正方案并不是没有缺点的，因为可以从上面的方案电路中看到，在设计的顶层添加的top_reg触发器虽然能够保证其中的这个或逻辑输出的信号通过该顶层触发器安全的输出，但是，随之也就会引出这样的一个问题，对于这类顶层的触发器的时钟信号和复位信号该怎样去选取，这是在使用触发器不得不需要考虑的首要问题，而且这样的后果势必也会增加SOC的设计面积的增加，但是时钟信号和复位信号的选取对于顶层的设计者来说还是一件十分困扰的事情。

另外的，还存在另一种修正方案：首先可以考虑不在顶层进行组合逻辑的集成，而是采取将这两个信号直接送入目的时钟域模块，先分别的进行同步的处理（主要是因为其是单bit的数据信号），同步到目的时钟域之后，再将同步之后的数据信号做或逻辑的操作，修正方案的电路结构如下所示：

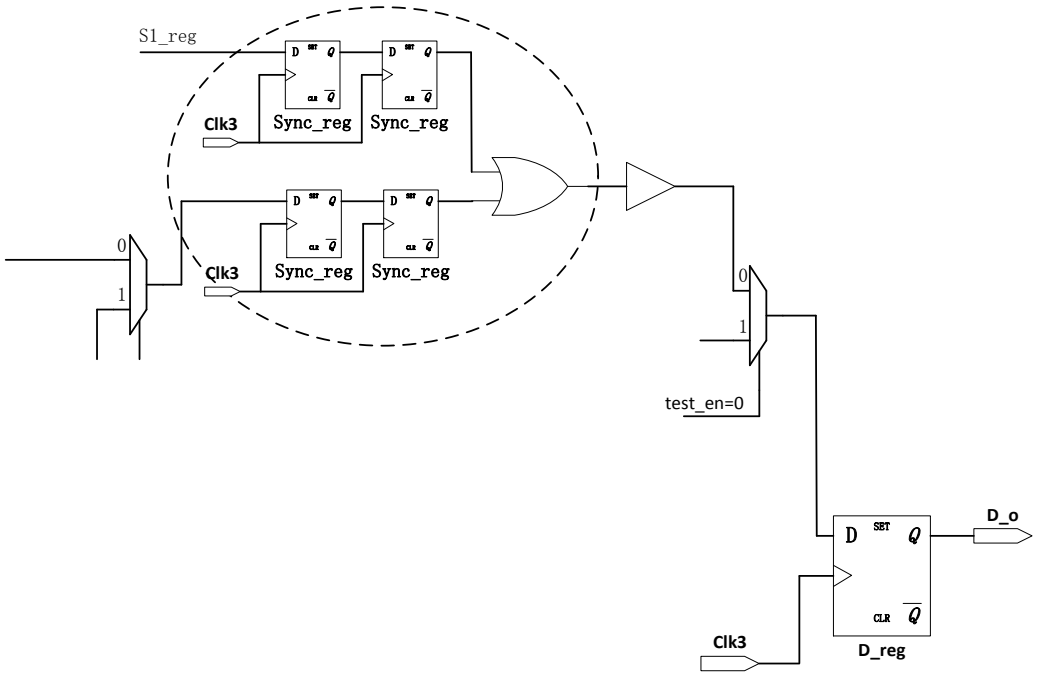


图 4.20 multi-sources 电路结构模型修正方案 2

对于上述的修正方案 2，可以看出，这种 RTL 方案由于使用了与目的时钟域保持一致的时钟信号和复位信号的同步触发器，这样的传输会更加的安全可靠。但是，也应该注意到这种方案将会大大的增加系统的时钟树和复位树在芯片中的

分布，并且也大大增加了芯片的设计面积，因此还是要结合两个修正方案进行权衡之后再去选择使用，以保证这样的电路结构和 CDC 传输路径尽量少的出现。

4.3.3 combo-logic 处于传输路径中的 violation 情况

第三种常见的violation信息，如下所示：

combinational logic used between crossings.

这种情况也是在顶层形成模块电路之间的集成的时候，经常能够碰到的设计结构。之所以会出现这种违例，从电路的结构来看，主要是因为，虽然在目的触发器实施了同步触发器，但是在传输的路径上存在着一写组合逻辑，这种组合逻辑在某些时候是非常致命的，因为其有一定的几率会产生毛刺，这些毛刺可能会被这些目的同步电路接收，一旦发生这种情况，后续电路的运转就会受到很大的影响。

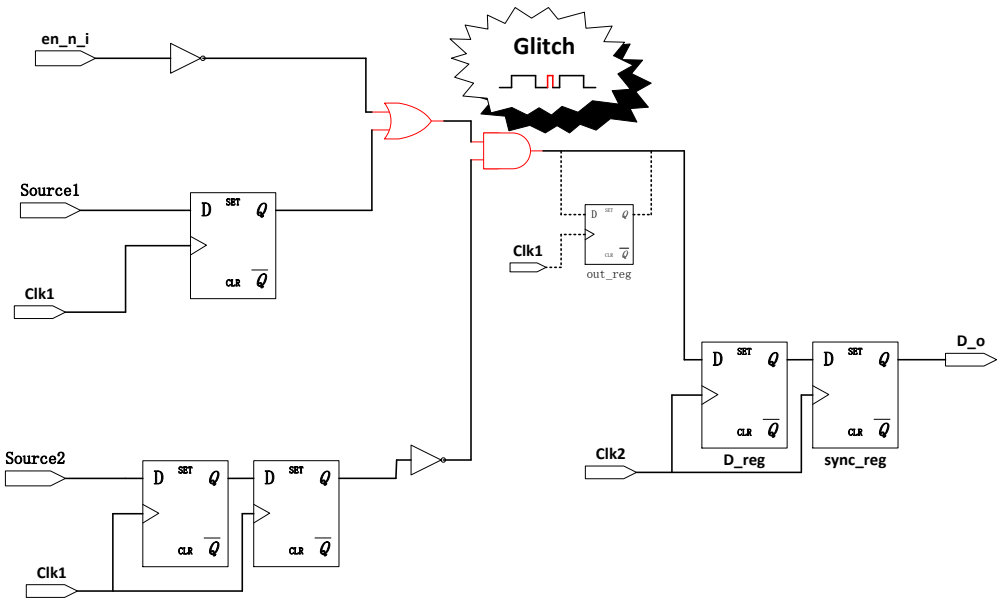


图 4.21 combo logic in the crossing 电路模型

上述的这个电路也是在实际设计中的一个简化的实例模型，其中可以看出，在图中或门处，有两个源时钟域的数据在这里进行了逻辑操作，虽然已经在目的时钟域实施了一个两级触发器的同步设计，但是由于组合逻辑非常容易产生不稳定毛刺的缘故，在长时间的运行中，一旦目的时钟域的同步设计触发器捕捉到这样的一个毛刺噪声信号，那么在目的时钟域将会看到一个单周期宽度的脉冲信号，如果同步触发器的后续电路中有对此类脉冲信号敏感的元器件的时候，例如常见的power_switch（电压开关）等，这样就会出现错误的逻辑，例如对于power_switch这样的开关来说，那就可能使其打开或者关闭开关，从而对整个系统的电流或者能耗造成影响。

因此常见的处理方法是：首先要排除其是否为静态信号或者可以被允许的逻辑结构，如果是，那么就可以在约束文件里面屏蔽掉该信号路径；否则常见的 RTL 修正办法如图中的虚线部分所示，需要加上一个寄存器，使其按照一定的时钟采样进行输出，从而去避免组合逻辑毛刺的产生。

4.3.4 目的触发器驱动多条数据路径的 violation 情况

第三种常见的 violation 信息，如下所示：

destination instance is driving multiple paths。

往往在进行目的时钟域的同步设计的时候，传统的触发器串联形式的同步设计也会用作一些逻辑的输入，一旦出现这种电路结构，系统可能就要面临传递亚稳态的可能性。SpyGlass 在静态分析的时候，也会对此类电路结构情况进行检查。

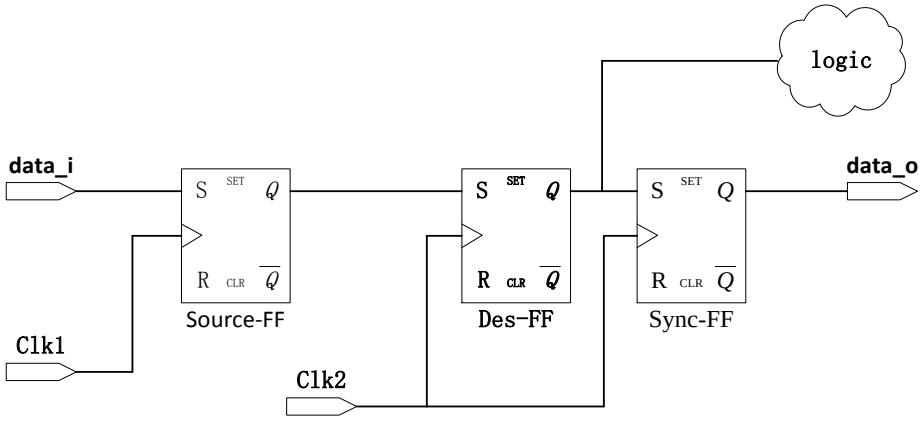


图 4.22 Des-FF driving multi paths 电路模型

通过图 4.22 电路模型看出，不管是这个 Des-FF 触发器后面驱动的是何种类型的电路结构，都有可能形成和传递亚稳态的情况出现。在进行 SOC RTL 设计的过程中，也不免会碰到这样的情况出现。特别是在进行边沿或者脉冲采样的电路应用中。

对于这种类型的违例信息以及电路结构，首先要考虑的是此条传输路径上所传输的信号类型是不是允许这种结构的使用，如果允许则进行 waive 处理；其次要判定目的和源的时钟信号连接于设计规范一致，因为通常要进行边沿或者脉冲采样的时候，采样的第一个数据都必须是同步设计输出的，如果排除此情况则需要 RTL 进行修正始终连接关系；最后就要开始考虑通过修正 RTL 完善该同步设计触发器来保证后续逻辑的输入信号是稳定安全的。简单的处理就是在 Des-FF 之前插入一级触发器。

4.3.5 hold-time check failed 的 violation 情况

第三种常见的violation信息如下示例：

data hold check: failed

在一个示例设计中，所涉及的两个时钟信号：clk_2s_i是源时钟域的时钟信号；clk_2d_i为目的时钟域的时钟信号。但是从时钟的相位周期定义上面来说，目的时钟域的一个周期内将会有两个源时钟域的时钟周期，这样就形成了第二章所述的CDC中异步输入数据的保持时间问题。SpyGlass的一个检查结果如下所示：

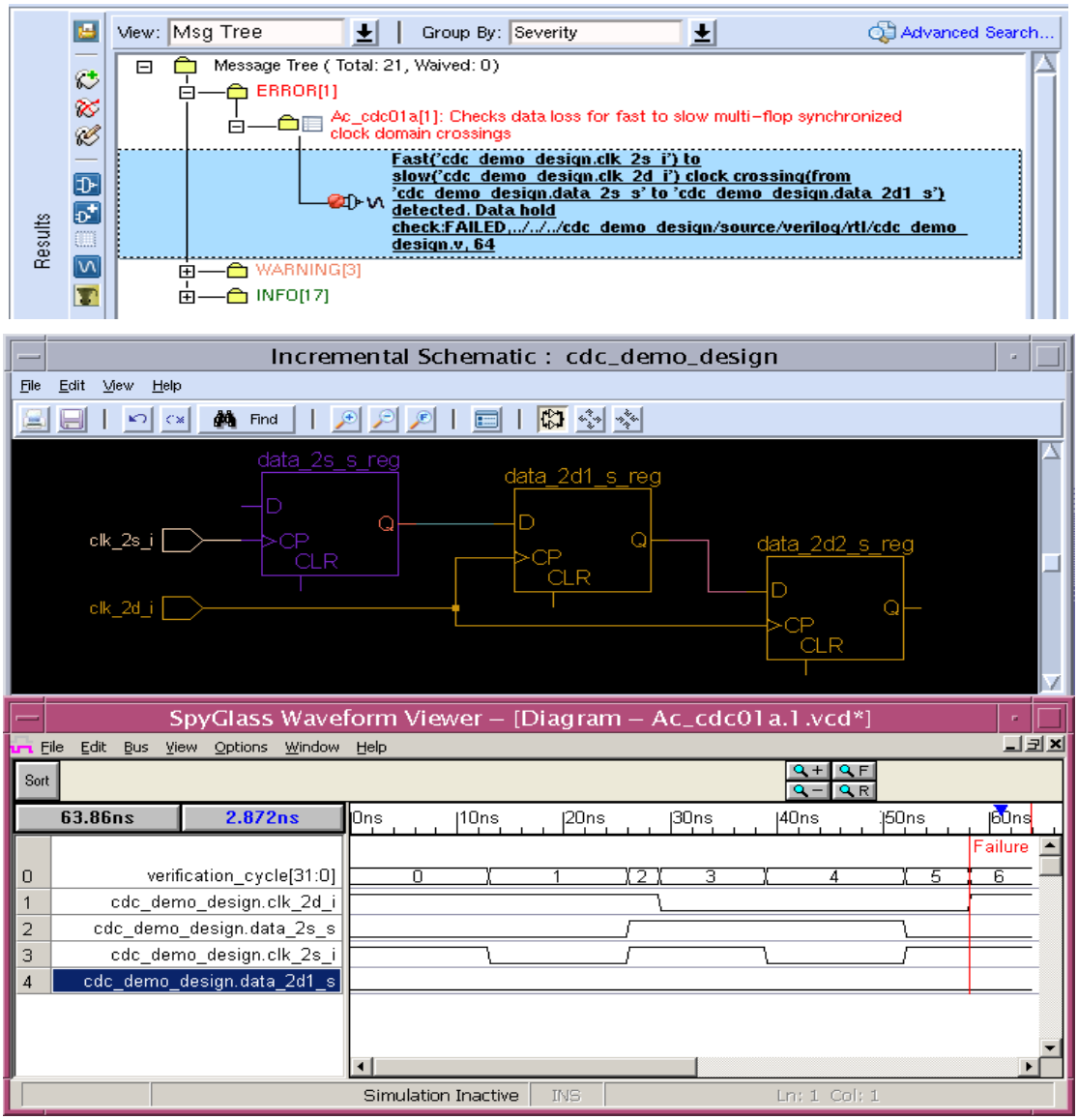


图 4.23 相关的 RTL 电路结构以及 SpyGlass 静态仿真波形

对于此种RTL的缺陷，我们常见的修正方案有两种，其中一种是我们在第三章讲述的慢时钟域到快时钟域的脉冲同步设计，根据实际的情况我们决定使用无反馈型的脉冲同步设计或者带busy反馈型的脉冲同步设计；另外一种则是设计比较复

杂的握手协议的同步设计。当然，也许对于此类的设计违例，我们应该还需要考虑是否是时钟信号的定义与设计规范一致。因此，所有的设计修正方案，都应该遵守满足现实设计应用的需求。

SpyGlass 所能够静态检查出的不合格的同步设计的电路结构还有很多种，上述的这几种情况是通常在设计的工作中经常见到的类型，对于单 bit 的数据信号容易引起毛刺，而对于多 bit 数据则主要会容易引起后续控制电路或者 FSM（有限状态机）时序的错乱。

4.4 本章小结

通过前面对于传统验证方法在 SOC 设计流程中对 CDC 传输路径以及 CDC 同步验证的诸多缺陷和弊端，本章主要设计并介绍了基于 SpyGlass 的 CDC 静态验证方法。

对于目前大规模数字 SOC 设计，必须有一种新型的、专门的 CDC 验证方法。其能够识别以上所有的同步设计类型，包括用户自定义的同步设计。并且能够在设计的较早阶段完成对 RTL 级的 CDC 验证工作，为后续的流程提供安全、干净的 RTL 设计。这里引入了基于 SpyGlass 的 CDC 静态的验证方法，其能够很好的满足以上的需求，同时由于是静态的验证，所以其对于验证环境的要求也比传统的动态验证方法要低。

结合目前数字 SOC 项目的实际情况，设计并且搭建了基于 SpyGlass 的 CDC 静态验证的一个基本的环境和流程。包括对于可综合 RTL、hard macro（硬核）、soft macro（软核）、模拟部分的行为模块等不同代码类型的处理方法。同时结合常见的时钟模块的结构，定义所需的必要约束文件。同时，对于 SpyGlass 内部能够识别的一些特殊的电路模型也给出了简单的介绍。

最后依据在实际项目中的验证结果，给出了常见的几种 RTL 的 CDC 同步设计缺陷。其中包含所有没有同步设计的 CDC 传输路径、多源信号的汇聚路径、CDC 同步路径上存在组合逻辑、目的同步触发器驱动多条路径以及 SpyGlass 所具备的时钟频率关联检查、防止 CDC 数据丢失的基本仿真功能。同时对于这几种常见的 CDC 静态验证结果，根据实际的设计情况，分析并且给出了相关的 RTL 或者时钟等约束文件的修正方案。通过实际的项目检验，这种基于 SpyGlass 的静态的 CDC 验证方法还是可行的。

对于这种基于 SpyGlass 的 CDC 静态的验证方法，可以看出其能够检验出同步设计的存在性以及部分功能的正确性等两个重要方面。当然，如果这种静态的 CDC 验证方法能够和传统的功能性验证等相结合使用的话，其所达到的效果应该是最理想的。在目前的工程设计中，这种验证的趋势将成为一种必然。

第五章 总结与展望

5.1 总结

本文系统的论述了 CDC 传输机理以及常见的一些 CDC 问题，结合目前设计中常见的多种不同类型的数据同步设计模型以及异步复位信号的同步，通过详细的分析目前数字 SOC 设计流程中所存在的一些传统的验证方法，在 CDC 同步设计验证方面的缺陷。重点的介绍了一种在前端 RTL 级的、基于 SpyGlass 的新型的 CDC 静态验证方法。结合实际的数字 SOC 项目的架构，设计并且搭建了一套可行的验证环境和验证流程。同时对于常见的 RTL 验证结果给出了一些 RTL 以及约束文件的修正方案。通过实际项目的检验，此种方法能够很好的保证设计前端的 RTL 设计质量。经过上述的一系列的介绍，对于解决同类的 CDC 同步设计和验证问题，存在着一定的借鉴意义。

5.2 技术展望

通过上述的 SpyGlass 的这一系列的静态检查的介绍，对于这样的验证方法，其实还是能够看出其可以改进的一面。由于实际的数字 SOC 设计中，为了能够在设计的后期进行芯片内部的测试和追踪工作，保证芯片最终的质量。通常会在系统中加入一些测试或者 bist（build-in self-testing）的设计模式，而在进行 SpyGlass 静态分析的时候，都是在设计的正常功能模式下面进行分析的，如果要切换到测试或者 bist 模式下分析，验证人员就必须要再次修改时钟信号的约束文件，并且修改相应的测试 mux 的 case_analysis 设置，然后重新做一次验证步骤，对于像基带 SOC 这样的大规模的系统设计来说，这种做法势必会降低 CDC 验证的效率，因此对于将来的 CDC 静态验证方法来讲：应该能够同时允许多种模式的静态检查在一次的验证过程中就能够得到充分的验证；另外还应该改进当使用静态验证方法在检查 RTL 设计结构时候对组合逻辑的友好程度。因为对于大规模的数字 SOC 设计来说很有可能出现相当复杂的组合逻辑在 CDC 的路径上；还有除了我们所介绍的常见应用同步设计之外，是否还存在另外的多种同步形式能够适应越来越快、超低功耗的大规模 SOC 的设计要求。相信随着研究的深入，还是能够继续的在这方面取得突破性的进展[18]。

致谢

我首先要感谢能够在英特尔移动通信技术(西安)有限公司获得实习的机会，并且通过其实际的项目得到了很大的锻炼和收获，也为该论文工作提供了许多第一手的参考资料；在此，我尤其要感谢我的实习 mentor 郝朝龙高级工程师，他严谨、敏捷的思维和工作方式，教会了我许多实际工作的技巧。另外不仅在论文的选题上面给出了耐心地指导，同时在实际的项目工作中以及论文过程中也给予了我极大的帮助。

我还要感谢我的导师汤晓燕老师，她平易近人的品格，认真细心的态度，对我的论文写作给予了极大的鼓励和帮助。

最后，感谢在实习工作和论文写作过程中帮助过我的所有人!

参考文献

[1] Lee, Hin-Kwai. Methodology for verifying multi-cycle and clock-domain-crossing logic using random flip-flop delays. US patent. Oct 2007

[2] Real Intent Inc. Clock Domain Crossing Demystified: The Second Generation Solution for CDC verification. Sunburst Design, Inc.2007

[3] CDC methodology cookbook: CDC Sub-Methodology User Guide. Page13 to 19 Atrenta, Inc.2012

[4] Clifford E. Cummings Clock Domain Crossing Design & Verification Techniques Using System Verilog. Sunburst, Inc.2008

[5] R.Ginorsa, “Metastability and synchronizers: A Tutorial,” IEEE D&T, 28(5): 23-35, 2011

[6] S. Beer, R. Ginosar, J. Cox, etc., “Metastability challenges for 65nm and beyond: Simulation and measurements,” Design Automation and Test in Europe (DATE), Grenoble, France, March 2013

[7] S. Beer, R. Ginosar, M. Priel, etc., "The Devolution of Synchronizers," ASYNC 2010

[8] S. Beer, R. Ginosar, R. Dobkin, etc., "MTBF Estimation in Coherent Clock Domains," Nineteenth IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), Santa Monica, USA, May 2013

[9] S. Beer and R. Ginosar, “Supply Voltage and Temperature Variations in Synchronization Circuits,” 2013

[10] Shengxian Zhuang2, Weidong Li, Jonas Carlsson. An Asynchronous Wrapper with Novel Handshake Circuits for GALS Systems. Linkoping University.2002

[11] San Jose. Real Intent Introduces CDC Verification for Altera Customers. Real Intent Inc. 2008.3

[12] Clifford E. Cummings , Simulation and Synthesis Techniques for Asynchronous FIFO Design Sunburst Design, Inc.2002

[13] Peter Alfke. Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons. Sunburst Design, Inc. Xilinx, Inc.2003

[14] Michelle Lange. Automated CDC verification protects complex electronic hardware from metastability failures. Mentor Graphics Corporation. Aug. 2008

[15] Ping Yeung, Ph.D. Five Steps to Quality CDC Verification. Mentor Graphics.Inc.2007

[16] R. Dobkin, T. Kapshitz, S. Flur and R.Ginosar, "Assertion Based Verification of Multiple-Clock GALS Systems," VLSI-SoC, Oct. 2008

[17] Sanjay. Churiwala, Sapan Garg, Verification of Clock Domain Crossing in SoCs. http://chipdesignmag.com/display.php?articleId=3303

[18] Vaishnav.Gorur, Lending a ‘Formal’ Hand to CDC Verification: A Case Study of Non-Intuitive Failure Signatures, Real Intent, June 14, 2013