知乎

干货大放送之CDC工程经验总结--CDC的那些事（7）完结篇

李虹江

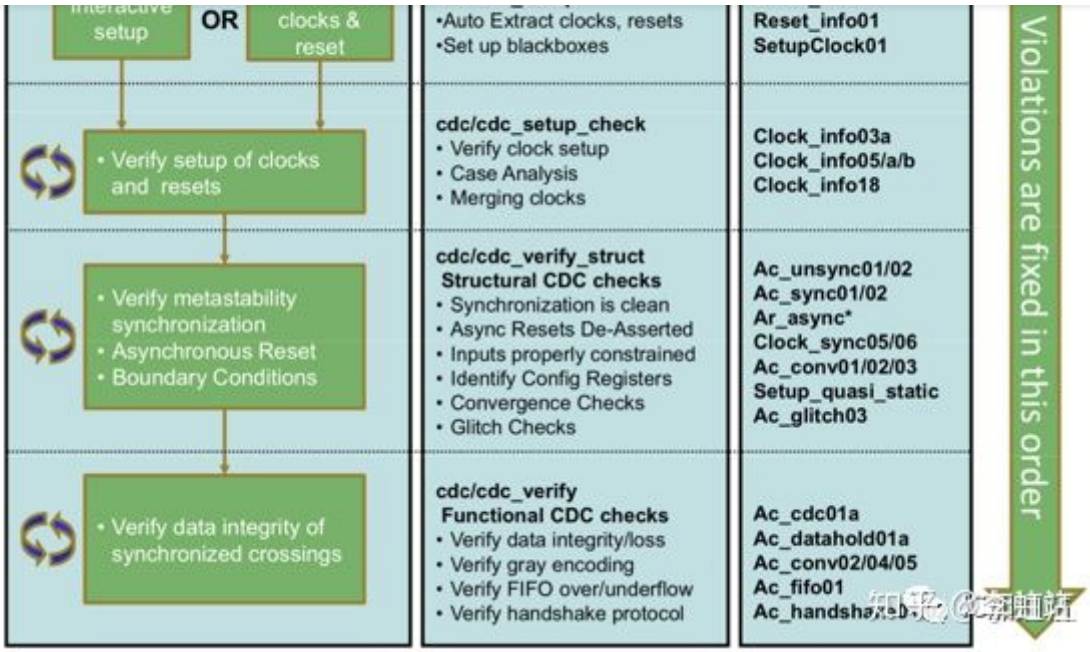
硅谷老李，微信公众号“IC加油站”

这一篇老李给大家简单介绍一下工业界常用的CDC检查工具Spyglass，然后奉上CDC设计和验证中的工程经验总结。如果你已经熟悉Spyglass CDC，那么你可以跳过第一部分。全篇干货满满，总计三千多字，希望大家一定能够读到最后，欢迎点赞以及分享朋友圈。

一、CDC检查工具

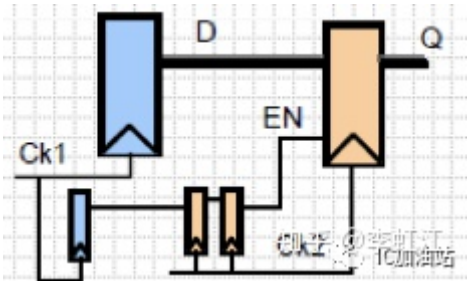
我们先来说CDC检查工具。业界三大EDA公司Synopsys, Cadence, Mentor都有各自的CDC工具：Synopsys Spyglass CDC®, Cadence Conformal Constraint Designer®, Mentor Questa®. 而这其中以Synopsys公司的Spyglass CDC最为常用，市场占有率最高。在很多公司里都以Spyglass CDC作为Sign-off的标准。

Spyglass CDC的基本功能是它已经定义好了一系列的设计规则(rule)，然后基于这些规则来对设计进行检查，看设计是不是满足了这些规则，如果有违反规则的设计，它会把这些违反的地方报告出来让设计人员进行debug。不同的规则被划分为几个大类，称之为goal。举例来说，首先它会对设计和constraint进行设置检查(setup rule check)，比如是不是每一个flop的时钟都有定义，reset信号有没有定义等等。当setup rule check过了之后，它会进行一系列的CDC检查，比如是否有信号跨时钟域但是没有经过synchronizer等等。每一个rule下面所有的不满足规则的设计它都会报出来，可以通过schematic, spreedsheet, text report的形式呈现给设计人员，同时会有详细的规则介绍以及为什么它认为这个地方的设计违反了该设计规则，并且给出了相应的修改建议。



上面这个图列出了Spyglass CDC的检查流程，从setup开始，进行setup check，之后再进行structure check，最后再进行functional check。structual check是静态的，工具会直接分析设计的有没有满足基本的跨时钟域规则，但是工具是不理解这个设计的功能是什么。而functional CDC check 需要更进一步理解设计的功能，从实际功能运行的角度进一步分析设计，找出设计中的缺陷。

绝大部分情况下，structure check就可以找出大多数的CDC问题。那为什么还要进行functional check呢？举个例子，如下面的电路



Q flop是有一个enable的，这个enable 信号是来自clk1，并且用2flop synchronizer进行了同步，在structure check中，这个enable信号会被当做qualifier，工具会认为这是设计者有意为之，并不会报错。因为很可能设计者的意图就是利用这个enable信号来杜绝D信号的CDC问题。回想一下，我们在讲多bit信号同步里面其实就是用这个思路的：即同步一个单bit的load信号到clk2去，只有load信号为1时，Q才会采D的值。但是注意，利用这种方法来同步有个假定：即在clk1

和其他检查工具一样，我们不仅要给工具提供我们的RTL design，还要提供一个constraint file来告诉工具一些design的基本信息，比如clock，reset都是什么，以及告诉工具一些分析CDC时要用到的信息，这样产生的最终分析结果才能有效。Sypgass CDC的constraint file我们称之为sgdc。相信很多人已经了解sdc， sdc全称是synopsys design constraint, 主要是定义design的clock, clock relationship，case analysis等等，主要用在综合和STA工具里。对于CDC检查来说，光有clock的定义还不够，通常还要有reset相关的信息，而且用作timing分析的case analysis以及false path并不能直接适用在CDC的check上，这些信息都要在sgdc里提供。

在structure check里，大家最常见的violation有以下几种：

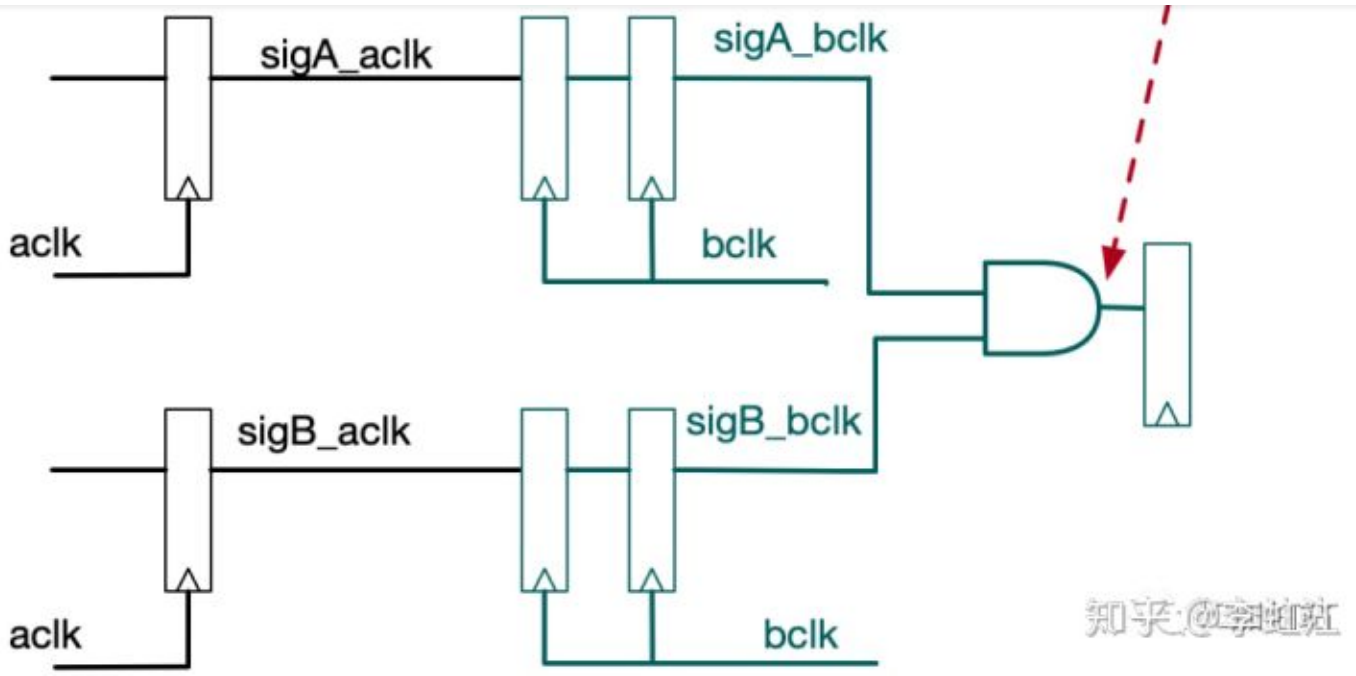
Ac_unsync01/02

Ac_unsync简单来说就是该同步的信号没有进行同步，这是CDC里最重要的rule，所有Ac_unsync的violation都不应该waive，而是要仔细分析design，来给信号加上适当的synchronizer。Ac_unsync分为以下两个子类：

- Ac_unsync01 ： 一个单bit信号跨时钟域，但是工具发现并没有进行同步
- Ac_unsync02: 一个vector即多bit信号跨了时钟域，但是工具没有发现进行同步

Ac_conv02

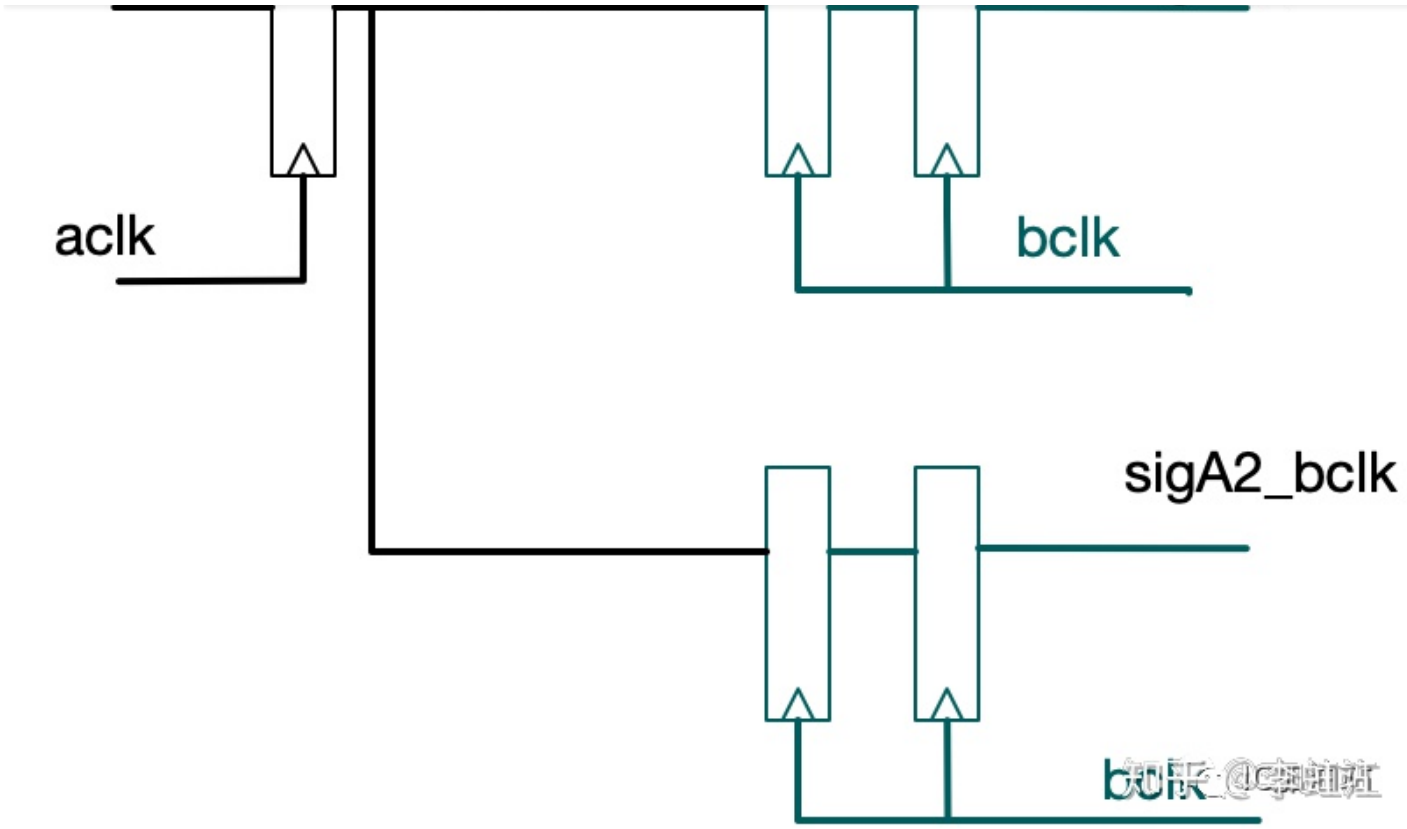
Convergence Violation也是在CDC分析中非常常见的一类violation。设计人员通常会记得给每个跨时钟域的信号加synchronizer，但是很多人一旦加完synchronizer之后就以为万事大吉，在destination clock域对信号随便进行操作了，从而容易产生convergence的问题。举个简单例子，单bit信号sigA和sigB，用2flop synchronizer同步到bclk域，然后进行AND操作，会有什么问题呢？其实问题和多bit信号利用2flop synchronizer产生错误是一个道理: 由于2flop synchronizer有随机性，当sigA_aclk和sigB_aclk同时变化时，sigA_bclk和sigB_bclk可能会出现不期望的组合。



有些时候当tool报告了Ac_conv02，但并不是必须要修改design。比如说如果设计本身可以保证sigA_aclk和sigB_aclk不会在同一个cycle变化，即它们本身满足gray code的特性，那这个电路其实是没有问题的。我们可以给CDC tool加gray_signals这个command来告诉工具这里没有问题。当然，每一条Ac_conv02的violation都要仔细分析。

Ac_coherency06

Ac_coherency06也是一个常见的violation，指的是同一个信号在同一个时钟域里被同步了多次。特别是design变得比较大，有多个设计人员来做一个系统，不同人负责不同模块的时候容易出现这个这个问题。因为可能你看到一个信号到了你设计的时钟域，你加了synchronizer，而这个信号可能还去了别人的模块，别人也加了synchronizer，但是你不知道，而你们两个模块恰好又是同一个时钟域，这个时候对整个大的系统进行分析的时候就会报出这个violation。（注：这个violation Spyglass CDC默认是warning, 不是error，但是还是建议大家看这个violation）CDC有的时候就是这么有意思，你不同步不行，你同步多了也会出问题。



首先，由于用了多个synchronizer产生了浪费，其次，依然是由于synchronizer的随机性，导致很可能sigA1_bclk和sigA2_bclk在不同的cycle发生了变化，相应的后级电路也可能因为它们变化不一致导致出错，因为从clean design的角度，既然是同一个信号，那么就不应该不一致。这种violation通常也比较好修，拿掉多余的synchronizer就可以了。

Ac_glitch02

Ac_glitch02也是非常常见的violation，原因就是synchronzier的input不是来自于flop，而是combo电路的输出。这个老李在你真的懂2-flop synchronizer吗-- CDC的那些事（2）里已经讲过了。

当然，Spyglass CDC还包含了其他上百个rule，更多的细节有待大家在工作中自己去学习，看文档掌握。

二、CDC工程经验总结

以下经验都是老李在工作中向前辈学习到以及自己总结的经验，每一条单独拿出来说都不复杂，希

我们在设计之前，其实要问自己，这一个功能我能不能不跨时钟域就实现呢？或者仔细想一想能不能将逻辑简化呢？比如可以将多bit信号简化为单bit信号再同步来省下一些synchronizer， 还要注意看是不是在别的地方已经对相关的信号进行了同步来避免coherency的问题。提前想好为什么要跨时钟域并进行恰当的设计，要远比在设计后期来一个一个去修CDC的问题要有价值的多。

- 一个module包含一个clock domain

在设计时，尽量把相同时钟域的逻辑放在同一个module内，这样对这个module可以认为是synchronous的设计。老李比较喜欢把所有的synchronization logic放在一个单独的module里，这样做的好处就是CDC的分界很清晰，以后debug以及修cdc的问题就只需要改这一个module了。

- 同步时不能根据source clock的频率和destination clock的频率关系来同步，而是要假定它们之间的关系是任意的。这样才能保证设计的健壮性和鲁棒性。
- 要例化已经验证过的CDC synchronizer library，比如2flop synchronizer, pulse synchronizer, async FIFO等等，成熟有一定积累的设计公司都已经有这些模块，在你的设计里直接例化它们，千万不要自己在RTL去造轮子。使用这些library对于其他flow来说也有帮助，比如说STA检查中可以直接将synchronizer上的path设为timing的false path，这个时候可以直接利用synchronizer的module name把这些cell找出来，非常方便。
- 所有的IP的output必须来自一个register，禁止将组合逻辑的输出当做IP的输出。
- 一个信号同步到另外一个时钟域只能同步一次，禁止同步多次，否则会有coherency的问题。
- 将要跨时钟域的信号标记上clk，要选择个统一的naming style。比如req_aclk, req_pclk分别表示req信号在aclk和在pclk，这样可以使得code清晰可读。
- 在功能仿真验证中，test bench要对2flop synchronizer来deposit随机的值，来模拟flop在产生metasability之后可能稳定在0，也可能稳定在1，用这种方法能够发现一些隐藏得很深的bug。
- 对于异步的时钟，在testbench里要使它们的沿要错开，而不要让他们的沿对齐。特别是有的时候两个clock的频率相同，但是如果它们是异步的，更要让沿错开，或者说要随机化它们之间的相位差。
- 在Spyglass CDC的check中
  - 要首先fix所有的setup error，之后再看CDC violation。因为很可能CDC的violation是由于设置不正确造成的。
  - 当有大量的相似的CDC error出现时，要考虑是不是setup没有设置正确，比如clock定义错误，quasi_static没有define等等。
  - 能从设计上修改最好，尽量减少cdc的waiver。
  - 不要相信tool给你找的qualifier，要么你自己指定qualifier，要么把tool帮你找到的qualifier从而认为path没有问题当做是有问题来进行分析。
  - 不要waive Ac_unsync01/02的violation，一定要从design上来fix

  - quasi_static: 这样tool就会认为这个信号是不会toggle的
  - qualifier: tool可以利用这个quailifer来去看multi-bit的signal是不是被这个qualifier来控制住从而不产生metastable。
  - cdc_false_path: 如果并不存在这样的functional path，比如一个信号去了DFT的逻辑但没有被synchronize。
  - 当然要慎重使用quasi_static, cdc_false_path, set_case_analysis，除非你有充足的理由。
- 对于较大的SoC，如果直接对整个design来跑CDC可能需要很长时间，而且最后产生的violation太多，非常不便于debug，这个时候可以考虑利用hierarchical flow。
  - 对一些小一些的模块先进行spyglass cdc check，把violation都清干净之后产生出一个abstract model。abstract model其实就是一个sgdc file，它描述了这个模块的CDC相关的所有信息，比如所有的input output port是在哪个clock domain上的，是combo的输出还是flop的输出等等。Abstract model还包含了这个模块的waiver。
  - 在上层模块run cdc的时候读入下层的abstract model，好处就是可以节省run time，同时下层模块内部的violation也不会报出来，这样可以使得最终的report可读性比较好。
  - 当然abstract model也不是万能的，有的时候甚至会掩盖掉一些真正的问题，所以使用abstract model时要谨慎。

以上就是老李给大家带来的干货，这也是老李关于CDC的那些事的最后一篇。老李这里做一个目录，方便大家看之前的内容。

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）

多bit信号跨时钟域怎么办？-- CDC的那些事（4）

面试必杀技：异步FIFO（上） -- CDC的那些事（5）

面试必杀技：异步FIFO（下）-- CDC的那些事（6）

欢迎大家继续关注微信公众号IC加油站

知乎

之前几篇老李发现收藏数居然比赞同数多，既然你觉得这篇文章有用，不妨点个“赞同”，让更多的人可以看到~

编辑于 06-29

芯片设计　ASIC　数字电路

推荐阅读

知乎

3 条评论 ⇌ 切换为时间排序

写下你的评论...

再见钟情 09-20

谢谢

赞

一只数字设计菜鸟 09-22

那SpyGlass CDC是怎么判断一个路径是不是异步呢?

赞

李虹江 (作者) 回复 一只数字设计菜鸟 09-24

clock定义了之后工具会自动判断每个flop属于哪个domain，就可以判断出一个path是不是异步的了

赞