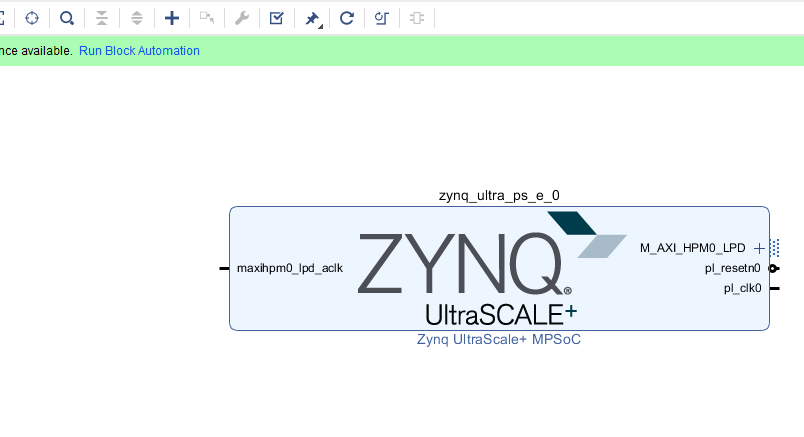
C2C指的是chip to chip，由于板子ps端的地址资源富足的离谱但是板上资源又不那么充裕，导致我们可以使用c2c这种操作来把板子连起来从而使得一个ps端占用多个板卡的资源。本教程内使用的是c2c的Aurora 链接方法，配合《板件连接模块.doc》使用效果更佳。

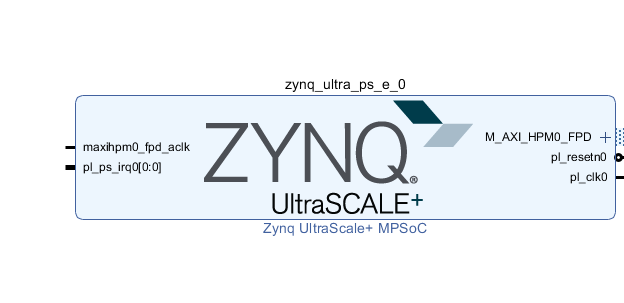
首先是总体结构，这个在板间连接那个文档中讲的很详细，在此就不再赘述了，直接进入工程实操部分。

本例程中使用最简单的ps->c2c->aurora->aurora->c2c->bram的结构，最终验证时使用devmem指令访问内存即可。

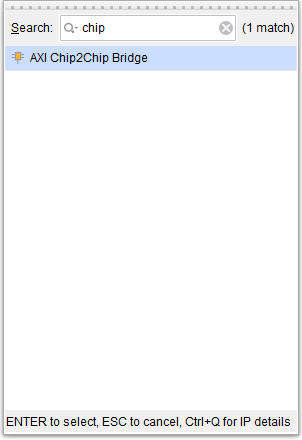
首先是master工程，该工程是主板工程，首先还是和之前一样选一个幸运路径新建一个平平无奇的ZCU106工程，先加入以下器件：

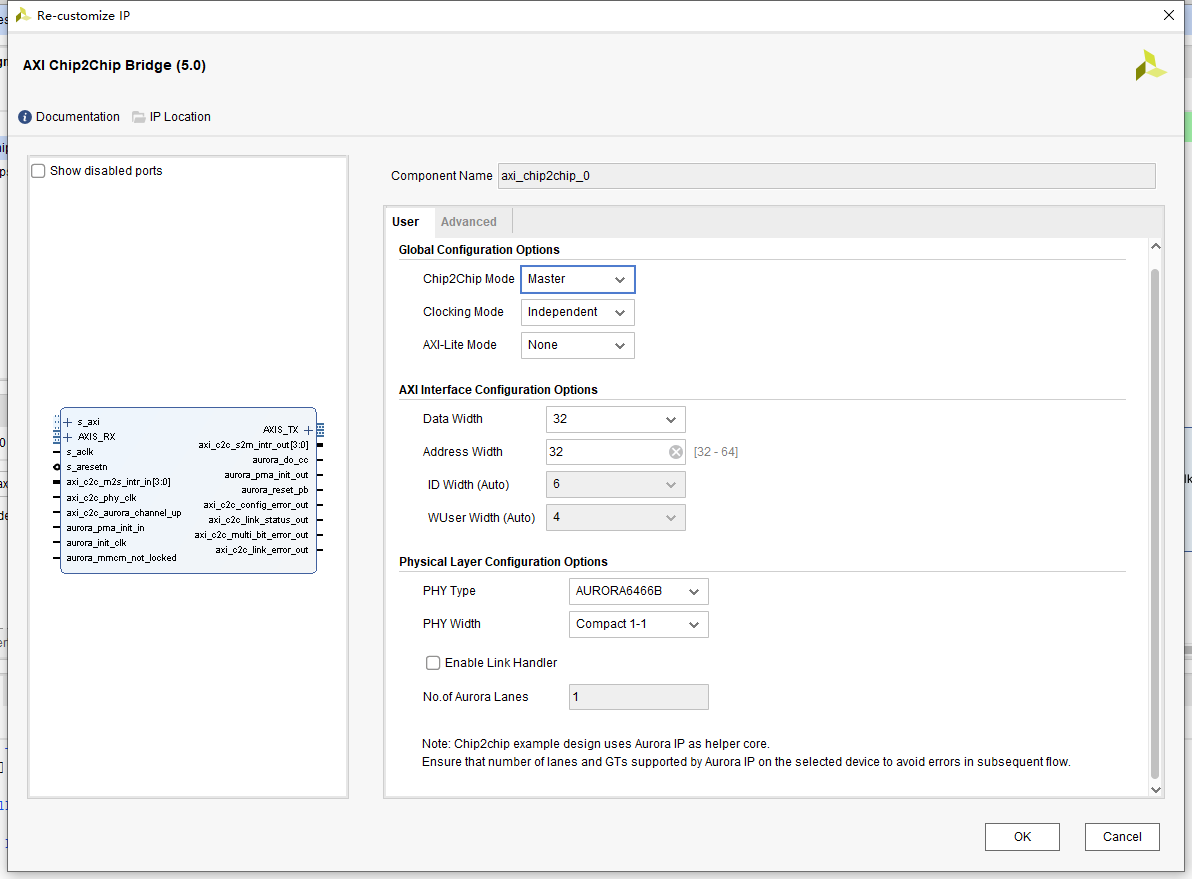


run block 然后再关掉一个fpd，反正最后弄到如下图所示状态即可



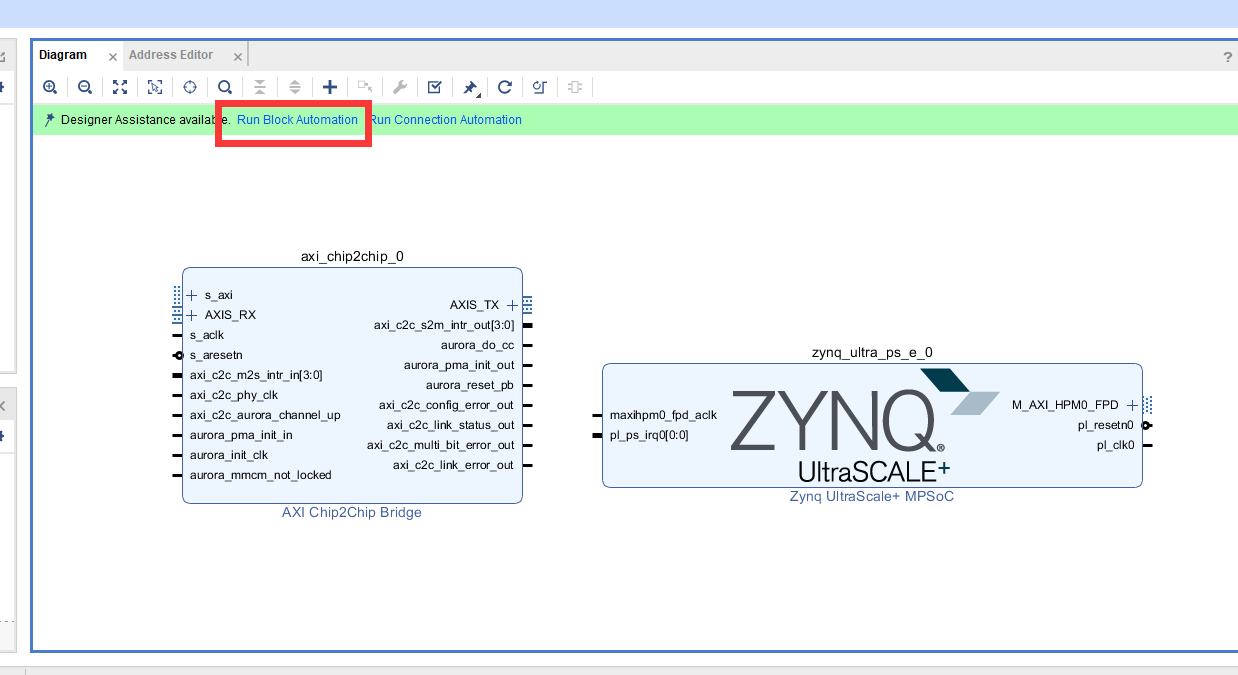
之后才是重头戏，添加



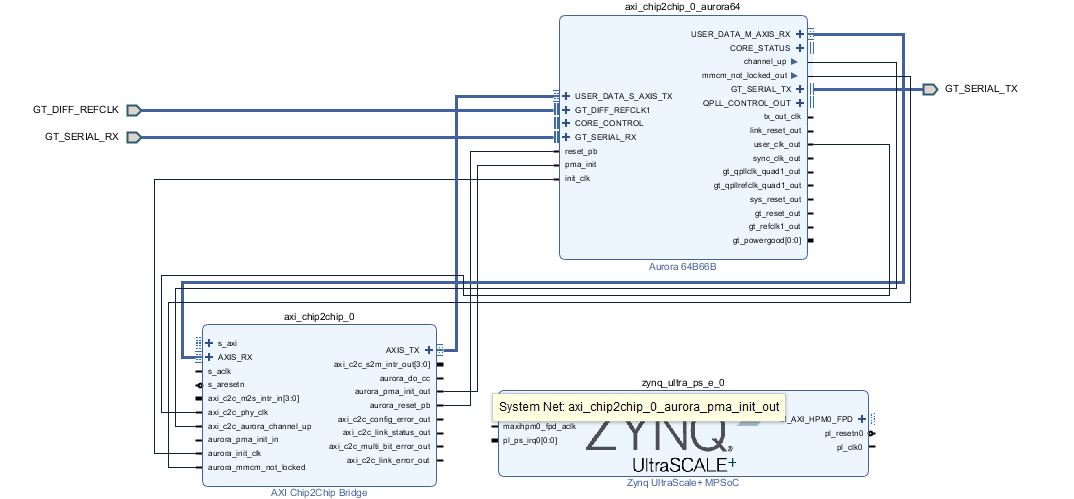


如图进行配置

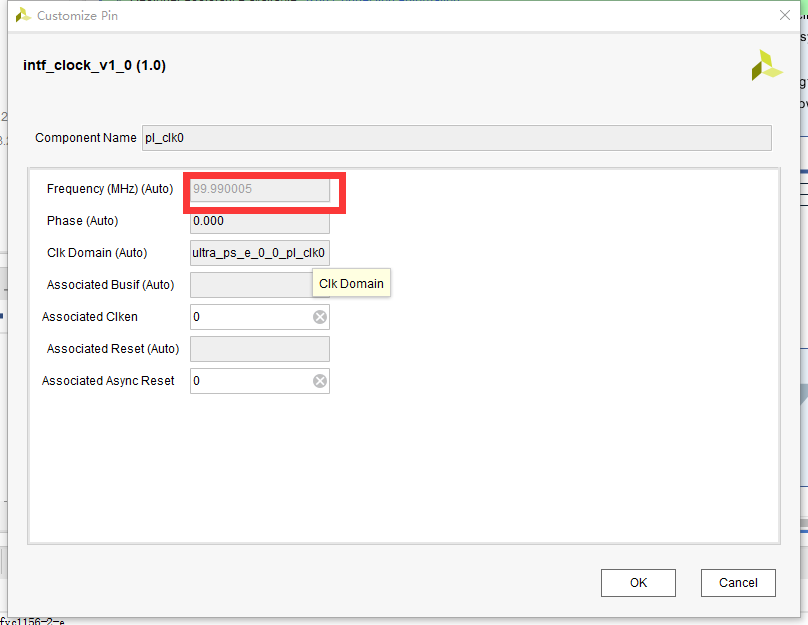
确定后多出一个如图所示的run block，点他



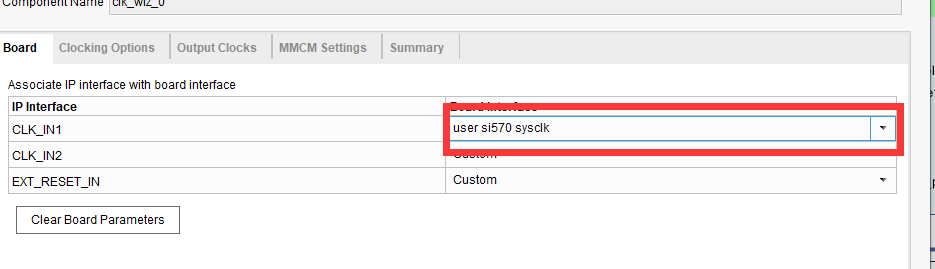
Run完你会发现它赠送了一个Aurora并且帮咱们连好了许多线

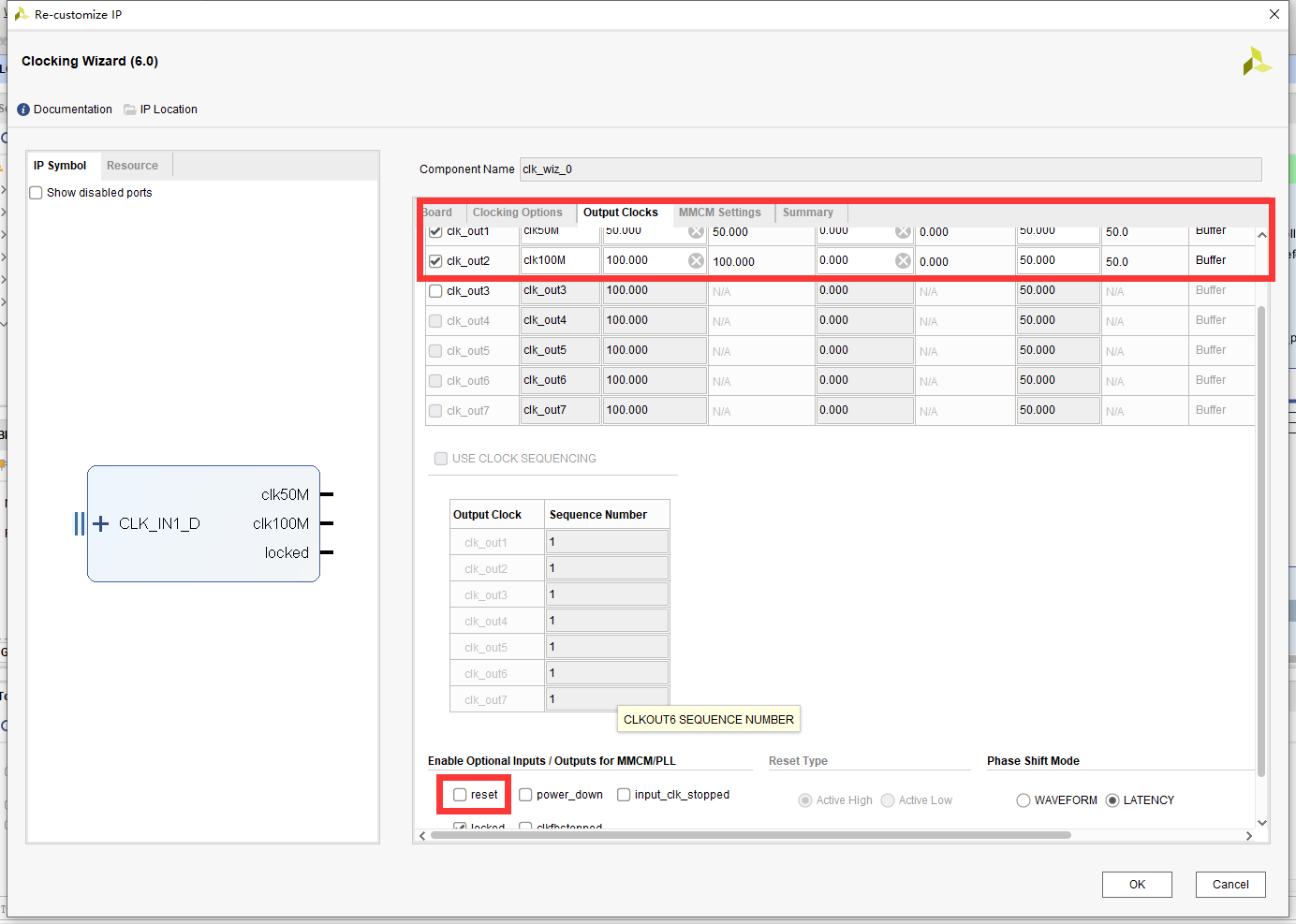


之后是一个有关时钟的问题，此时点击zynq上那个pl\_clk，你会看到它的时钟是



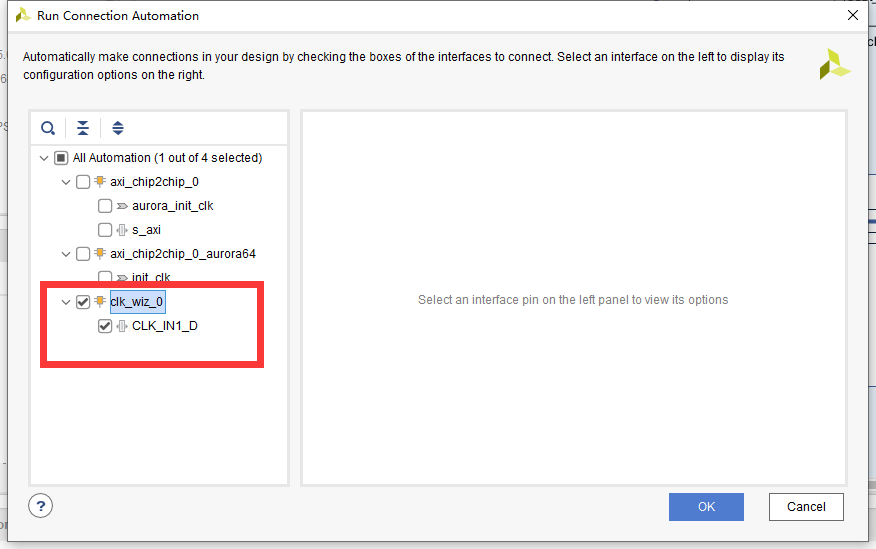
这吊数坑爹呢是，如果用这个直接控制axi时钟，从板为了与之对应也需要一个这个99.99…的时钟，这显然在难为clk wizard，但是zynq上那个M的axi口又必须得用他自己提供的pl时钟，所以我们用一个interconnect做一个时钟域隔离，让他隔离左边变成99…M，右边变为标准的100M，再加上给Aurora的初始化时钟，我们共定义两个时钟（为了区分连线），加入一个clkwiz，配置如下所示：



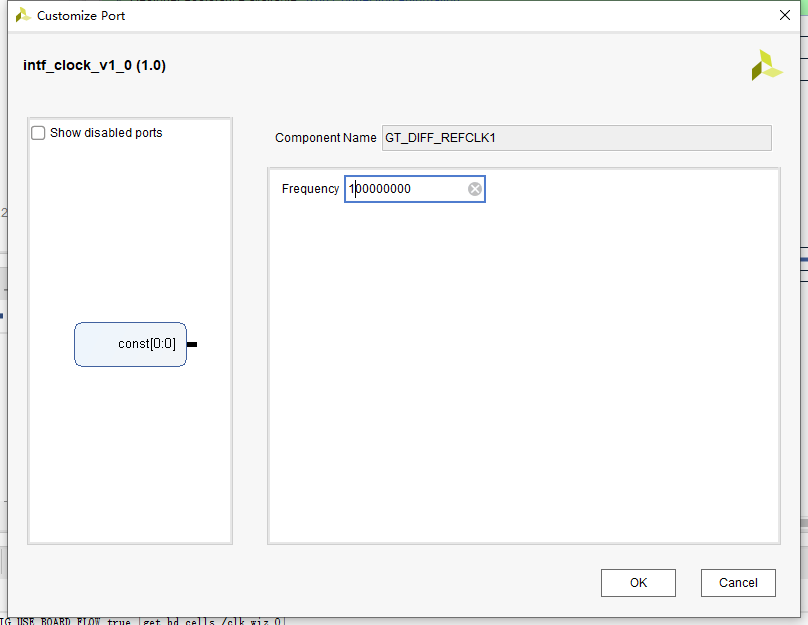


这里提个小细节，是昌钰师兄教的，在设置时钟时候把时钟的名字直接改成频率，这样连线时候巨清晰！！！（偶对了，如果是100.5这类的时钟改名字要改成100M5这种的，不能是100.5M，好像会报错）

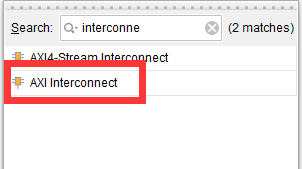
可以先点击一个run connection 只选择下图所示这块，把那个时钟接出来个头

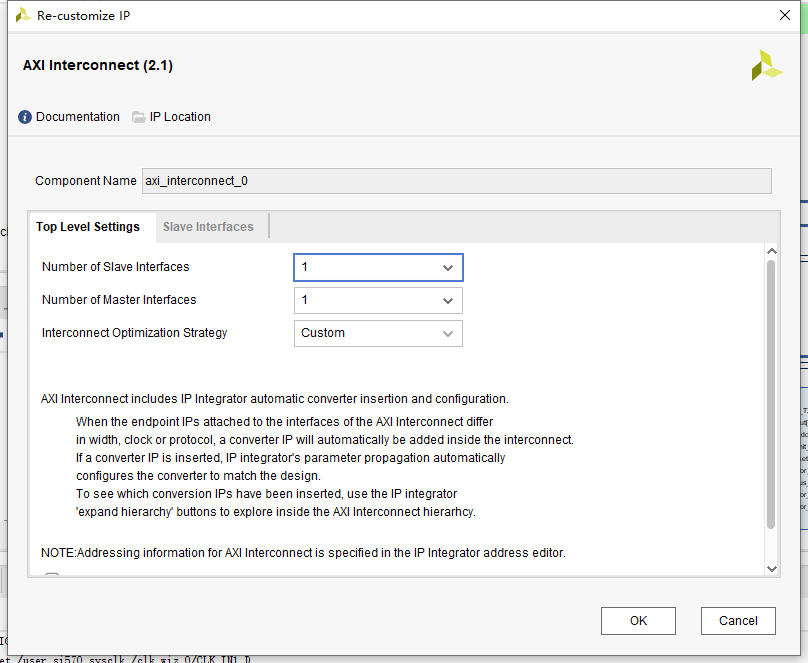


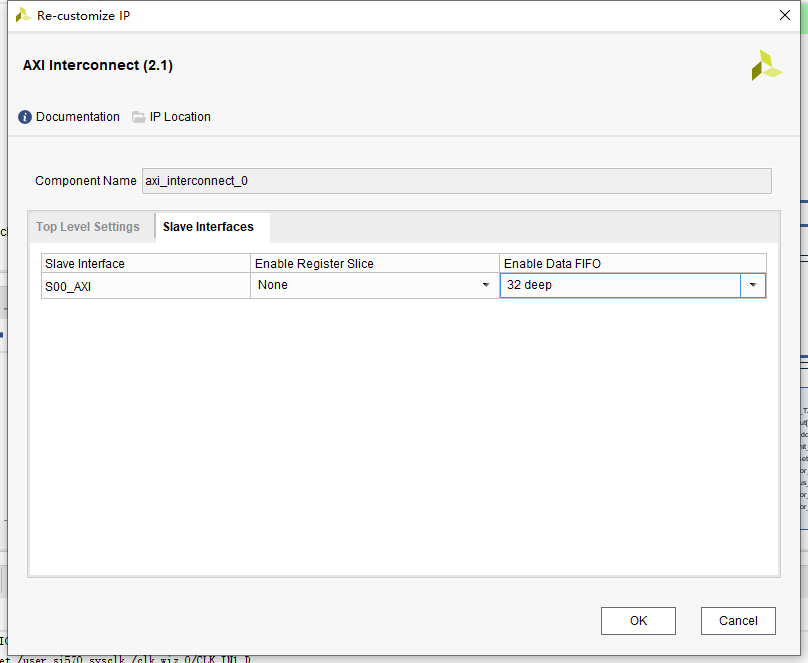
点多出来的那个头，把频率改了（跟后续选取板子上时钟管脚有关）



再加入interconnect，就可以开始着手连线了

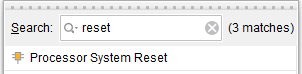




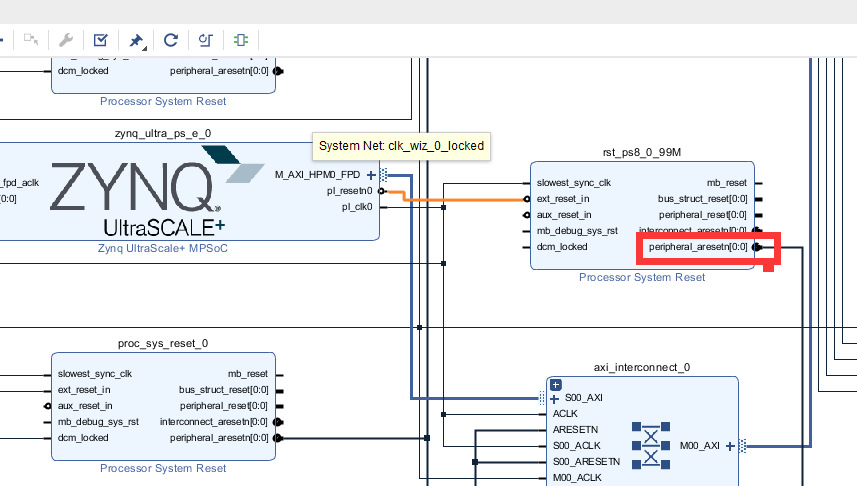


配置如上所示

Interconnect这个器件本身也需要一个输入时钟和reset，先给zynq配一个reset

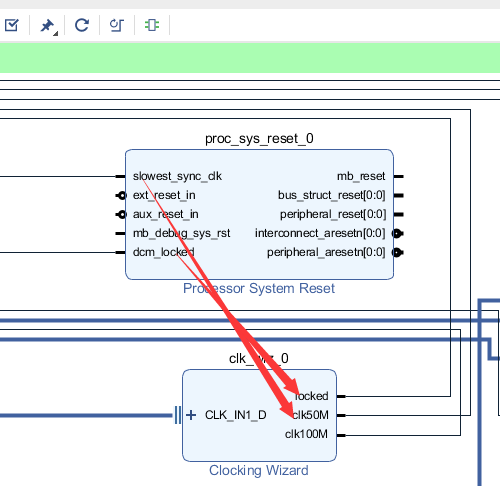


连线如下图



之后分别给axi-interconnect模块的输入时钟和s口时钟，输入resetn和s口resetn都连到pl\_clk以及红框内resetn上

在连M口之前，先为两个时钟加两个reset信号器件



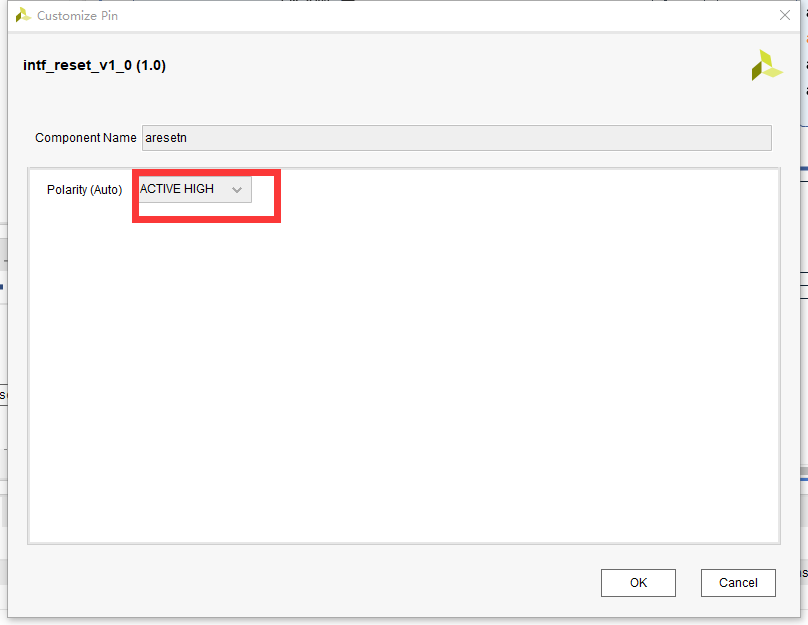
每个reset都如图连接（另一个reset的lock一样，时钟换成100M连slowest那个），之后看你要的器件是reset还是resetn来判断从哪个输出reset信号

M的时钟是我们希望和从板通信时使用的时钟，本次就采用100M，把时钟接过来

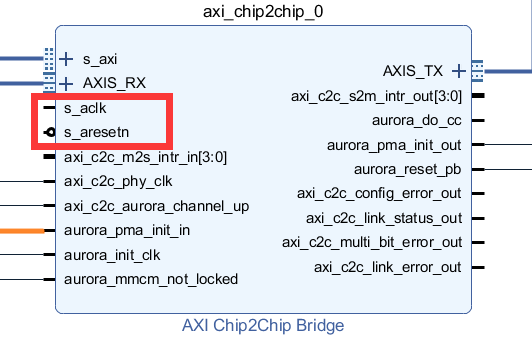
我们可以看到interconnect上是resetn，所以在100M的reset后面选perip\_reset接给M的reset

时钟域隔离部分至此结束

把右端的m口输入给c2c模块的s口，此次实验中我们选择的c2c和Aurora的初始化时钟是50m，把它连到c2c的Aurora\_init\_clk上并且为它输入reset信号，点击那个aurora\_pma\_init\_in,因为是

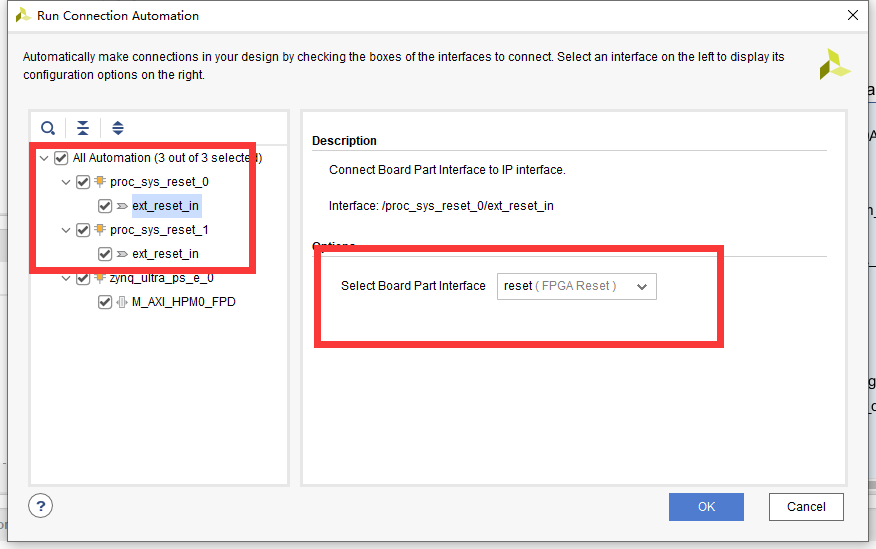


所以给它接50M的reset(不是resetn了)

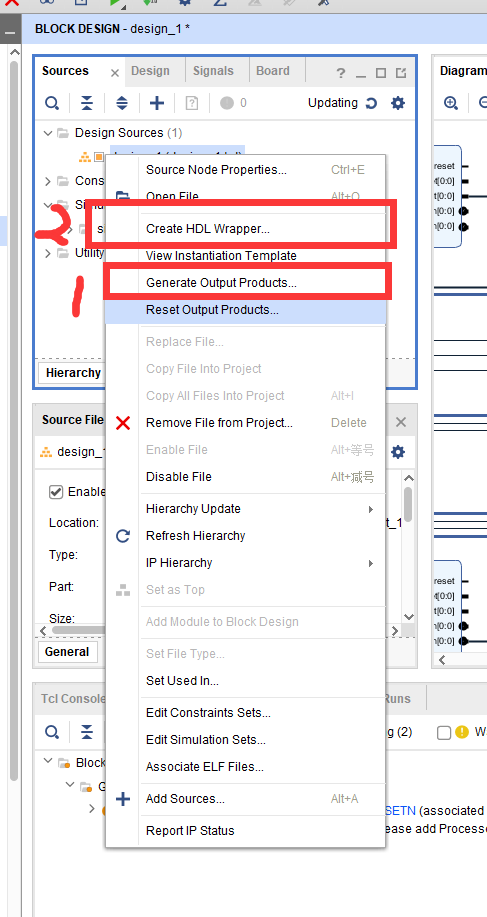


给这两个输入100m的resetn和时钟

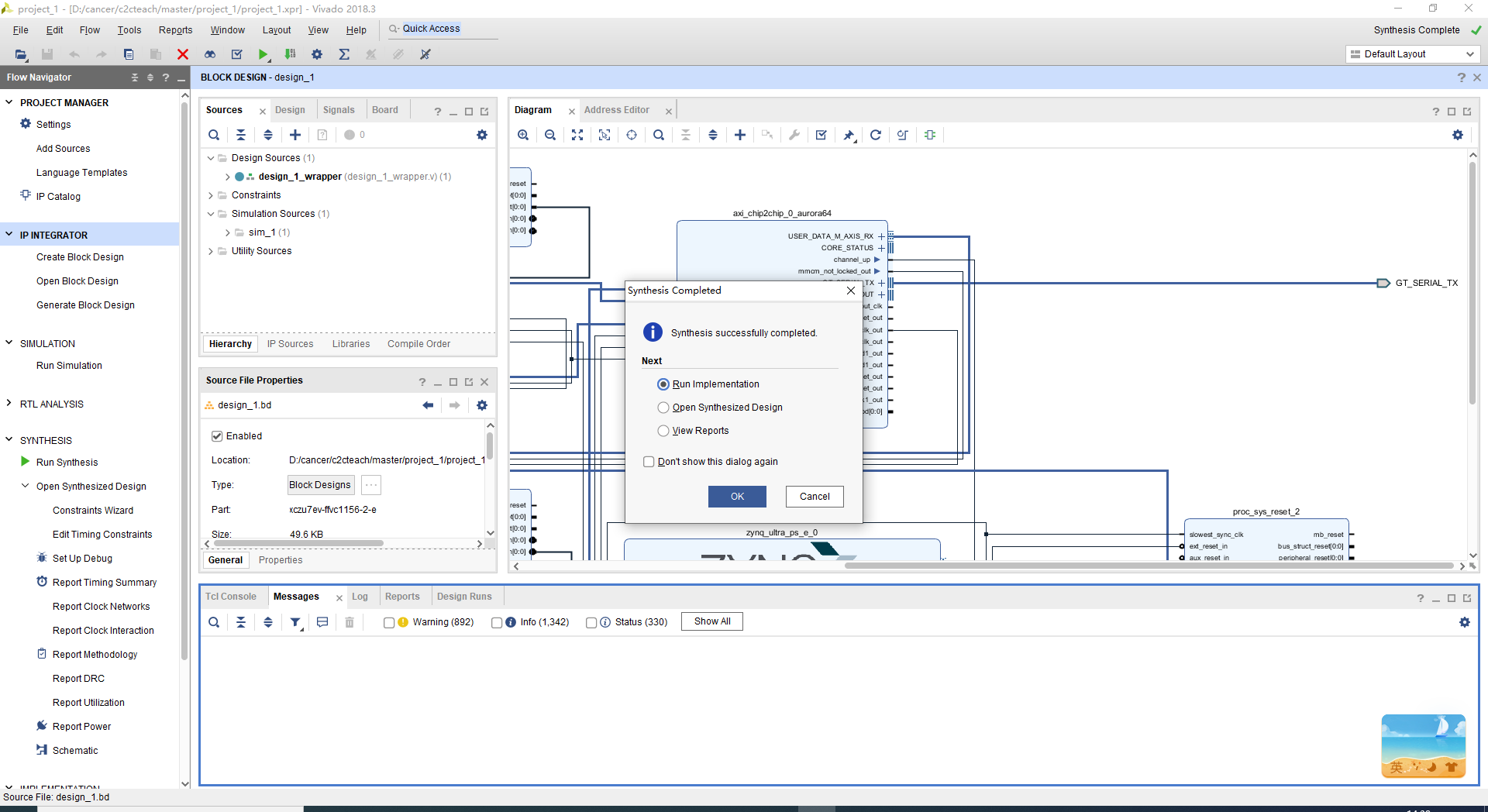
现在点击run auto connection，为红框里两个reset改下设置，剩下的不用改



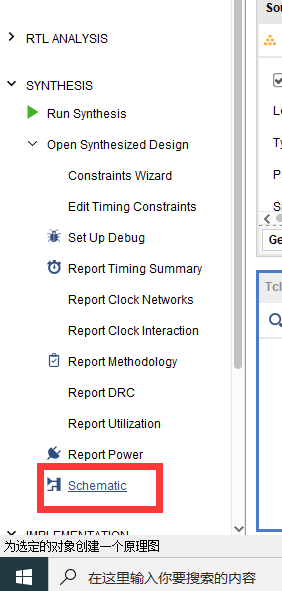
完成，验证一下bd，执行如图所示两步骤

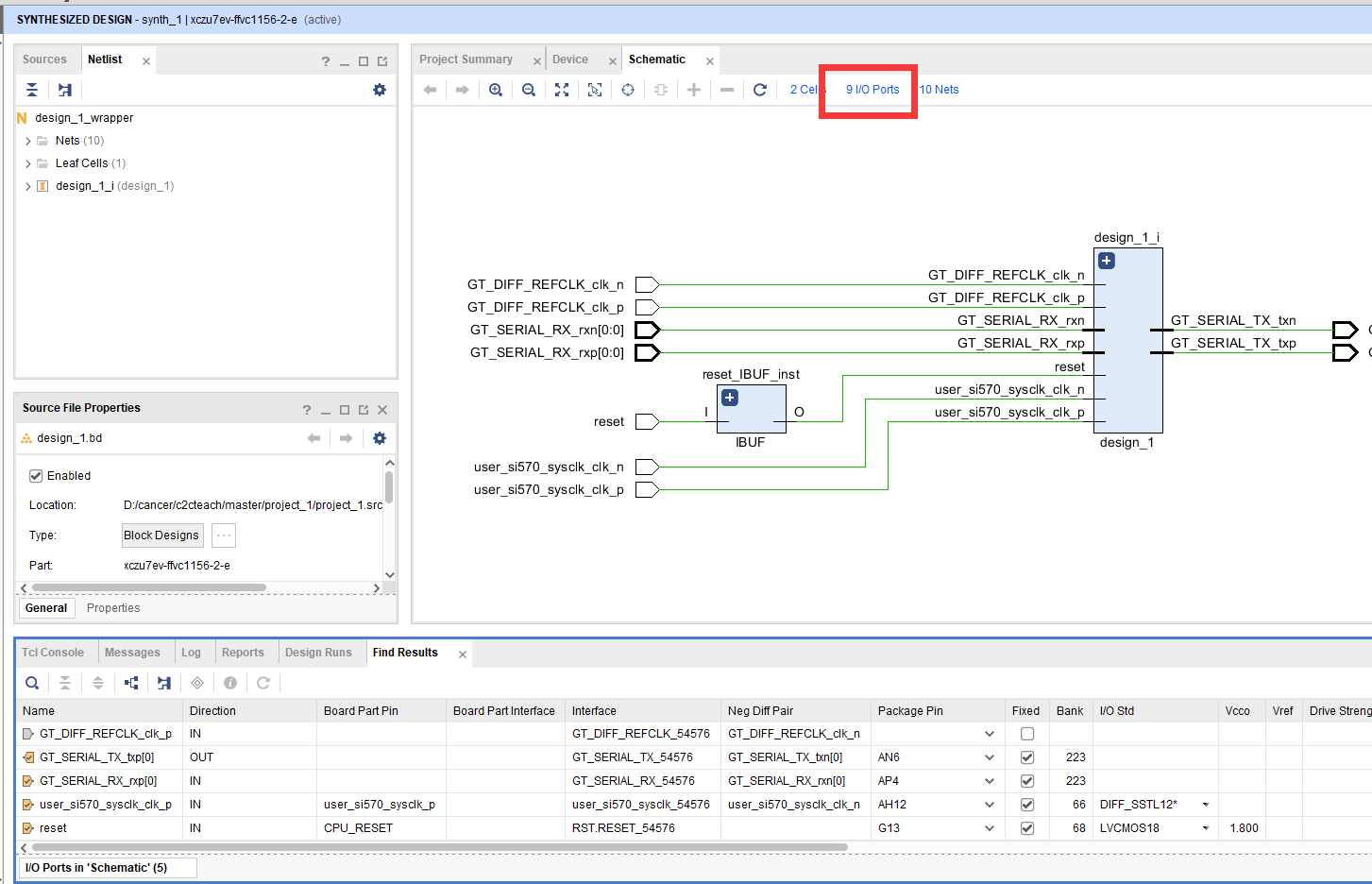


跑综合

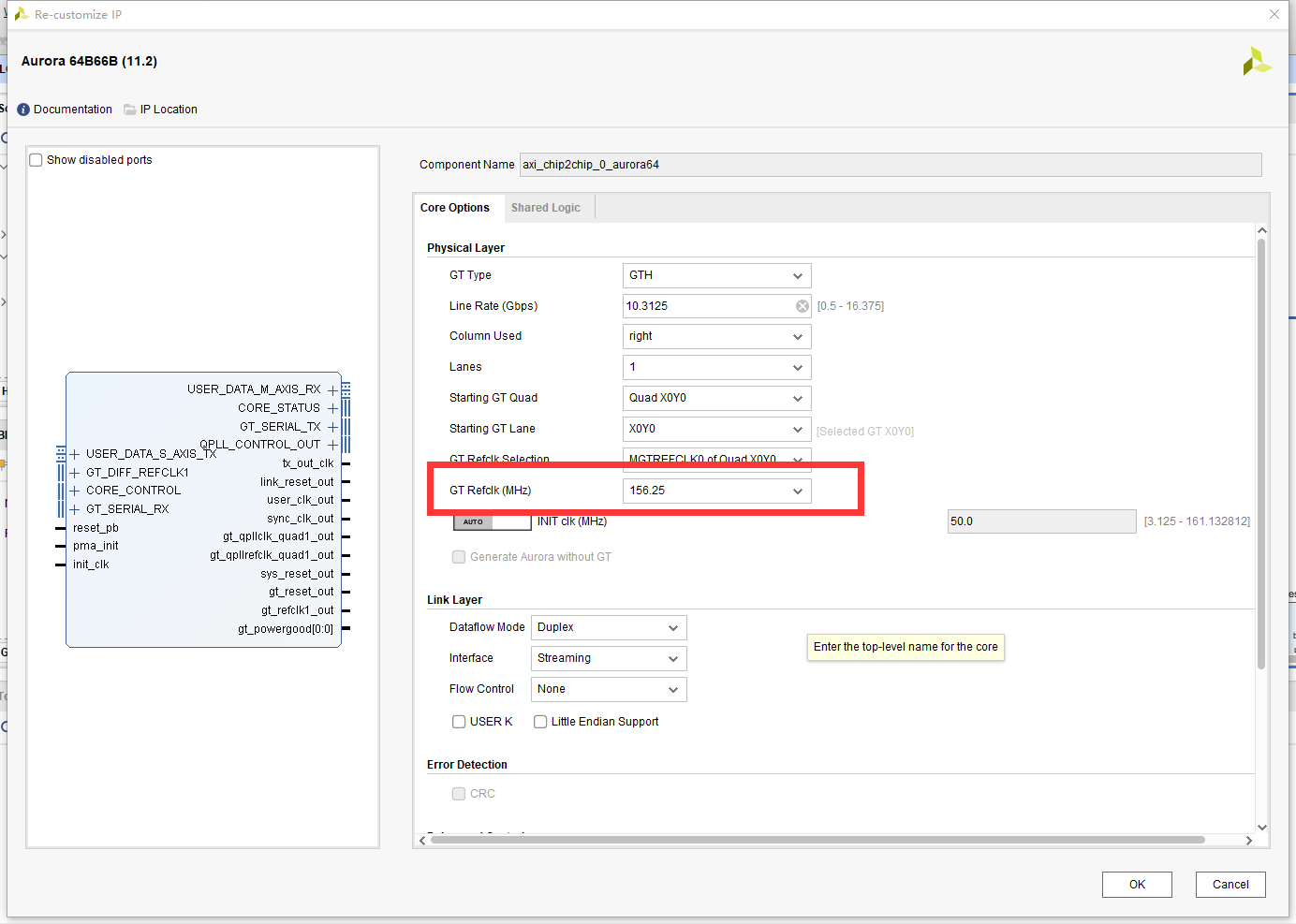


到这步点cancel调一下管脚

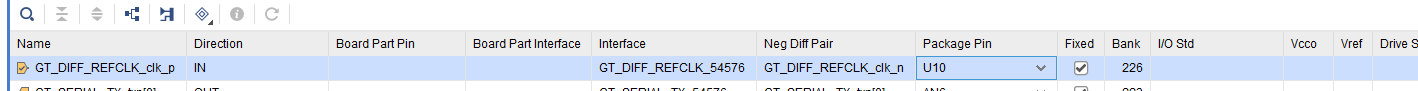




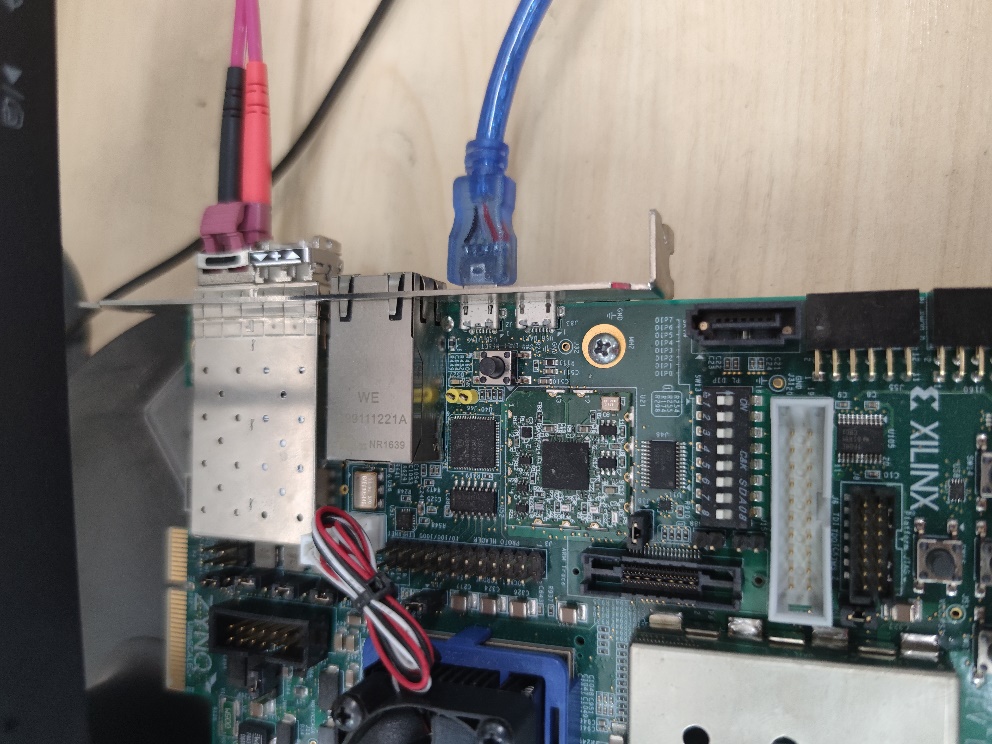
我们可以看到没有选中管脚的gtdiff时钟是灰色的，由于我们给c2c部分选的

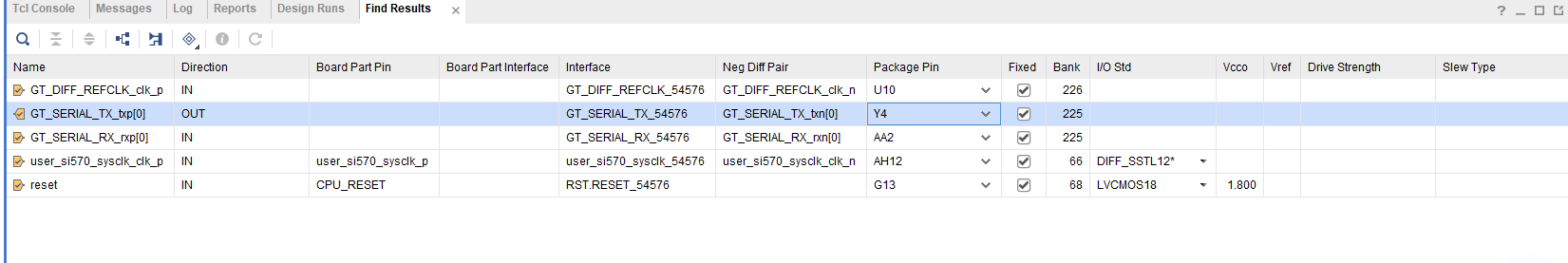


是156.25，所以选择U10（其实这个好像能查手册查到，先放着，后续再学习一下换了频率怎么查对应的）



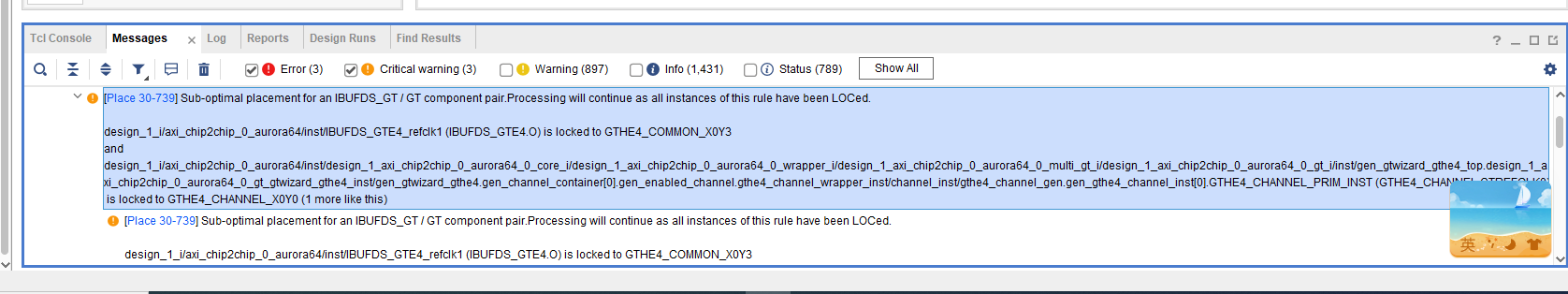
下面的改成ZCU106的板件连线0口对应的管脚（Y4和AA2处）

（最右上角那两根就是板间连线）



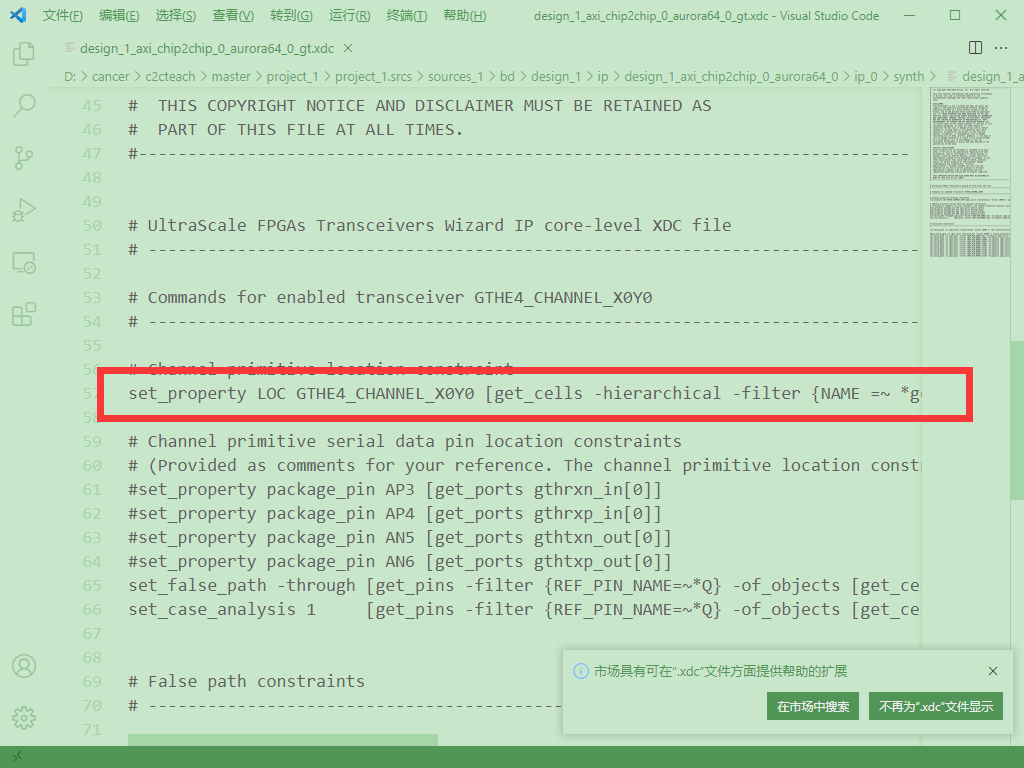
之后再跑实现

当我们满怀期望的等待结果时，估计会碰到如下的错误



不要慌张，打开如下所示路径中的文档（由于此处只有一个Aurora器件，所以只用改一个地方，正常是有几个Aurora改几个地方的）

D:\cancer\c2cteach\master\project\_1\project\_1.srcs\sources\_1\bd\design\_1\ip\design\_1\_axi\_chip2chip\_0\_aurora64\_0\ip\_0\synth\ design\_1\_axi\_chip2chip\_0\_aurora64\_0\_gt.xdc

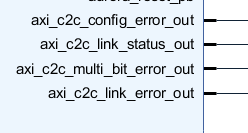


把这句倒霉孩子注释掉，再跑一遍就行了，具体原因好像是因为我们自行定义和vivado的冲突了

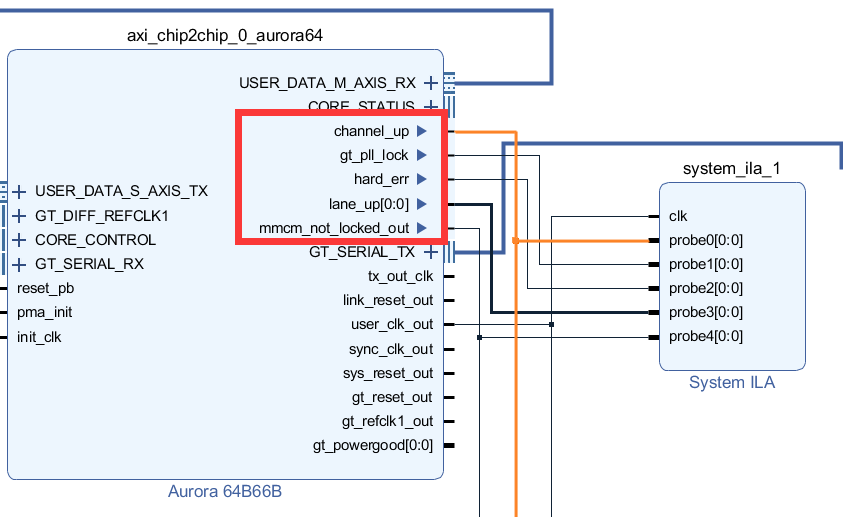
然后我们惊喜的发现，报错没了，诶起飞

然后生成bit流，主板vivado工程部分到此结束了，如果怕不能一次成想找找问题，可以增加ila进行抓信号，一般选择抓两个地方的信号：

第一个：

（在c2c板块上，因此给它的时钟是c2c板块的axi的时钟，本实验中为100M，其中axi\_c2c\_link\_status\_out信号拉高axi\_c2c\_config\_error\_out、axi\_c2c\_link\_error\_out信号置低。

第二个：



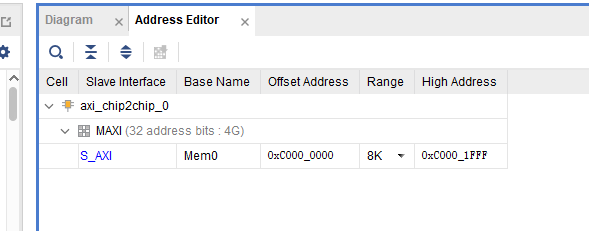
由于是Aurora的器件输出，所以我们直接用下面给的user\_clk\_out即可，Aurora IP上的lane\_up、channel\_up所有位上的信号均被拉高，mmcm\_not\_locked\_out置低表明物理通道联通且通道的驱动时钟稳定。把主板的工程弄到linux主机上做成petalinux备用即可

接下来是从板工程，建议选刚才的幸运路径下新建一个slave文件夹，两个工程放一起不会弄混。由于仅使用从板资源我们就不必加入zynq模块了，弄个c2c，和主板配置要完完全全一致（除了选slave）

加入老三样：c2c,clk wiz和axi\_bram

其他两个和之前一样的操作，bram改成单口，剩下和之前也一样，设两个reset，接接线就行了，唯一不同的是c2c上的m\_axi这次记得接到bram上，最后run connection，验证bd

需要注意的是



这个地方的起始地址要在之前主板工程的设计的范围内，要不整不了

之后的流程和前面也完全一致了。

把两个板子用线连起来，从板jtag模式，主板sd卡模式，连上串口（波特率115200），输入devmem 0xa0000000（你设置的地址中的一个）32，然后返回一个数，再devmem 0xa0000000 32 3，把3存进去，最后再看看存进去了没，返回数是3就通了

更正1：对于之前zynq出来的axi\_m信号只能用系统时钟：

其实是不对的，这个信号的时钟是



也就是说给这个口输入哪个时钟就随哪个时钟走，之前都是习惯性把pl\_clk连上去了，所以会涉及时钟域隔离的问题，理论上讲都用100M也可以