经验总结：FPGA 时序约束的 6 种方法

对自己的设计的实现方式越了解，对自己的设计的时序要求越了解，对目标器件的资源分布和结构越了解，对 EDA 工具执行约束的效果越了解，那么对设计的时序约束目标就会越清晰，相应地，设计的时序收敛过程就会更可控。

下文总结了几种进行时序约束的方法。按照从易到难的顺序排列如下：

0. 核心频率约束

这是最基本的，所以标号为 0。

1. 核心频率约束+时序例外约束

时序例外约束包括 FalsePath、MulticyclePath、MaxDelay、MinDelay。但这还不是最完整的时序约束。如果仅有这些约束的话，说明设计者的思路还局限在 FPGA 芯片内部。

2. 核心频率约束+时序例外约束+I/O 约束

I/O 约束包括引脚分配位置、空闲引脚驱动方式、外部走线延时（InputDelay、OutputDelay）、上下拉电阻、驱动电流强度等。加入 I/O 约束后的时序约束，才是完整的时序约束。FPGA 作为 PCB 上的一个器件，是整个 PCB 系统时序收敛的一部分。FPGA 作为 PCB 设计的一部分，是需要 PCB 设计工程师像对待所有 COTS 器件一样，阅读并分析其 I/O Timing Diagram 的。FPGA 不同于 COTS 器件之处在于，其 I/O Timing 是可以在设计后期在一定范围内调整的；虽然如此，最好还是在 PCB 设计前期给与充分的考虑并归入设计文档。

正因为 FPGA 的 I/O Timing 会在设计期间发生变化，所以准确地对其进行约束是保证设计稳定可控的重要因素。许多在 FPGA 重新编译后，FPGA 对外部器件的操作出现不稳定的问题都有可能是由此引起的。

3. 核心频率约束+时序例外约束+I/O 约束+Post-fit Netlist

引入 Post-fit Netlist 的过程是从一次成功的时序收敛结果开始，把特定的一组逻辑（Design Partition）在 FPGA 上实现的布局位置和布线结果（Netlist）固定下来，保证这一布局布线结果可以在新的编译中重现，相应地，这一组逻辑的时序收敛结果也就得到了保证。这个部分保留上一次编译结果的过程就是 Incremental Compilation，保留的网表类型和保留的程度都可以设置，而不仅仅局限于 Post-fit Netlist，从而获得相应的保留力度和优化效果。由于有了 EDA 工具的有力支持，虽然是精确到门级的细粒度约束，设计者只须进行一系列设置操作即可，不需要关心布局和布线的具体信息。由于精确到门级的约束内容过于繁多，在 qsf 文件中保存不下，得到保留的网表可以以 Partial Netlist 的形式输出到一个单独的文件 qxp 中，配和 qsf 文件中的粗略配置信息一起完成增量编译。

4. 核心频率约束+时序例外约束+I/O 约束+LogicLock

LogicLock 是在 FPGA 器件底层进行的布局约束。LogicLock 的约束是粗粒度的，只规定设计顶层模块或子模块可以调整的布局位置和大小（LogicLock Regions）。成功的 LogicLock 需要设计者对可能的时序收敛目标作出预计，考虑特定逻辑资源（引脚、存储器、DSP）与 LogicLock Region 的位置关系对时序的影响，并可以参考上一次时序成功收敛的

结果。这一权衡和规划 FPGA 底层物理布局的过程就是 FloorPlanning。LogicLock 给了设计者对布局位置和范围更多的控制权，可以有效地向 EDA 工具传递设计者的设计意图，避免 EDA 工具由于缺乏布局优先级信息而盲目优化非关键路径。由于模块在每一次编译中的布局位置变化被限定在了最优的固定范围内，时序收敛结果的可重现性也就更高。由于其粗粒度特性，LogicLock 的约束信息并不很多，可以在 qsf 文件中得到保留。

需要注意的是，方法 3 和 4 经常可以混合使用，即针对 FloorPlanning 指定的 LogicLock Region，把它作为一个 Design Partition 进行 Incremental Compilation。这是造成上述两种方法容易混淆的原因。

5. 核心频率约束+时序例外约束+I/O 约束+寄存器布局约束

寄存器布局约束是精确到寄存器或 LE 一级的细粒度布局约束。设计者通过对设计施加精准的控制来获得可靠的时序收敛结果。对设计中的每一个寄存器手工进行布局位置约束并保证时序收敛是一项浩大的工程，这标志着设计者能够完全控制设计的物理实现。这是一个理想目标，是不可能在有限的时间内完成的。通常的做法是设计者对设计的局部进行寄存器布局约束并通过实际运行布局布线工具来获得时序收敛的信息，通过数次迭代逼近预期的时序目标。

看到过一个这样的设计：一个子模块的每一个寄存器都得到了具体的布局位置约束。该模块的时序收敛也就相应地在每一次重新编译的过程中得到了保证。经过分析，这一子模块的设计和约束最初是在原理图中进行的，在达到时序收敛目标后该设计被转换为 HDL 语言描述，相应的约束也保存到了配置文件中。

6. 核心频率约束+时序例外约束+I/O 约束+特定路径延时约束

好的时序约束应该是“引导型”的，而不应该是“强制型”的。通过给出设计中关键路径的时序延迟范围，把具体而微的工作留给 EDA 工具在该约束的限定范围内自由实现。这也是一个理想目标，需要设计者对每一条时序路径都做到心中有数，需要设计者分清哪些路径是可以通过核心频率和简单的时序例外约束就可以收敛的，哪些路径是必须制定 MaxDelay 和 MinDelay 的，一条也不能遗漏，并且还需要 EDA 工具“善解人意”的有力支持。设定路径延时约束就是间接地设定布局布线约束，但是比上述 3、4、5 的方法更灵活，而且不失其准确性。通过时序约束而不是显式的布局和网表约束来达到时序收敛才是时序约束的真谛。

记得有人说过“好的时序是设计出来的，不是约束出来的”，我一直把这句话作为自己进行逻辑设计和时序约束的指导。好的约束必须以好的设计为前提。没有好的设计，在约束上下再大的功夫也是没有意义的。不过，通过正确的约束也可以检查设计的优劣，通过时序分析报告可以检查出设计上时序考虑不周的地方，从而加以修改。通过几次“分析—修改—分析”的迭代也可以达到完善设计的目标。应该说，设计是约束的根本，约束是设计的保证，二者是相辅相成的关系。