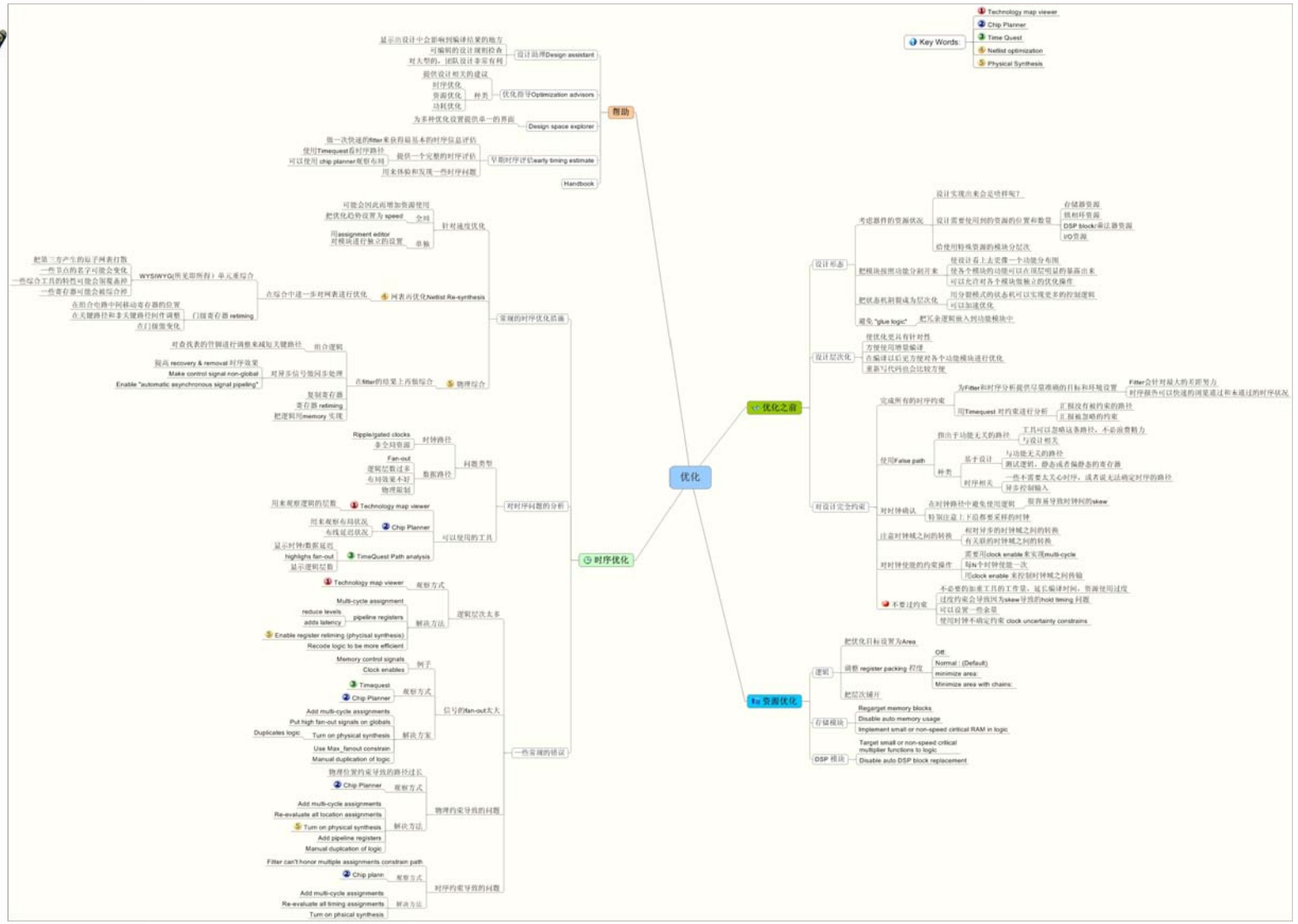
LEARNING THROUGH INNOVATION

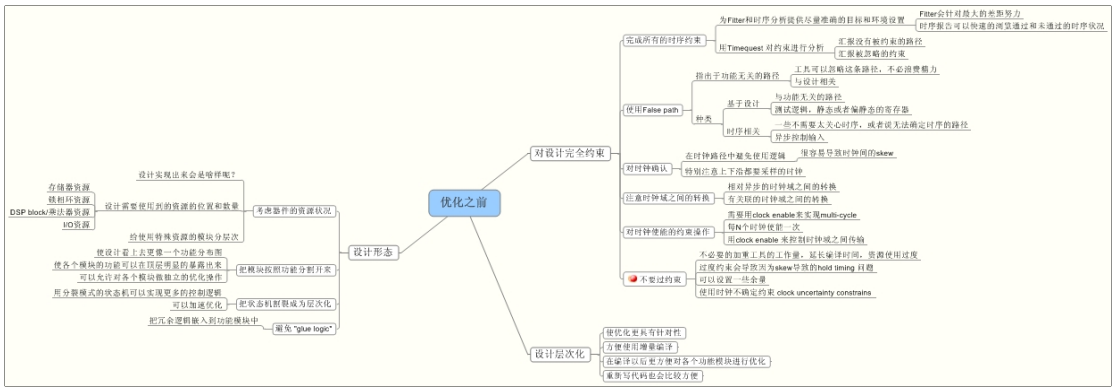
Innovate Asia

2008 Altera 亚洲创新设计大赛



8. 优化那些事儿

优化是一个很麻烦的问题，因为这个话题非常的杂，细碎。好像我们介绍一个人，你会一下子不知道从什么方面说比较好，因为他会同时拥有不同的身份，存在于社会当中。所以，我尽量的完成了这么一张图表，让大家对优化这个悬浮于平衡中间的一个游戏方法有一定的了解。平衡，其实是优化中的一个最关键的词。当然针对不同的应用，我们会有完全不同的需求，平衡点也会有算偏向。但是毕竟不能矫枉过正，太过苛求，否则只能是过犹不及了。这么说，似乎很没意思，让我们来说三个考量，时序，资源，功耗。这就是优化中的三个平衡极限。在有的设计中，算法对时序有要求，所以会要求设计能跑在一定的时钟频率上，这就需要对时序进行优化。有的公司在设计的后期开始考虑成本的问题，会希望选择尽量小的器件，那么这个时候，资源消耗变成了重点。而在手提式器件的设计中，功耗是至关重要的。而这三点，是没可能同时做到的。为了达到某种目的，你必须要付出其他的代价。所以，在做优化之前，做好一个优化目标是很有必要的。当然最基本的是时序，和资源。在这里我们比较重点的讨论这两方面的话题。 大家一定看到了前面这种让你晕得乱七八糟的图，我的任务，就是把他们解释一下。



- **优化之前**

在提优化之前，我们当然需要有一个提供优化的基本形态，就是你的设计。如果你的设计还没怎么完成，大可不必就着急的开始优化。因为每次编译都会把你的优化努力随机掉。而最好的优化方法，其实就是可以不优化。那就是把代码写的很优化，退而求其次，就是把代码写的容易优化。这里又要提老掉牙的事情了，代码要写得有层次化，好处就不罗嗦了。那么在写代码的时候需要考虑什么问题呢？

首先是你使用的目标器件的资源状况。通过一些小实验，你可以知道，你写出来的代码，大概会实现成什么样。这对你写代码有一种感官上的映射非常有帮助。然后就是一些特殊零件的数目和位置，比如存储器（memory），计算器（DSP block），特殊的管脚资源(LVDS)。

其实是把模块按照功能分割开来。从顶层电路看起来，真个设计就是一些功能模块的组合，看上去和规划的功能图一模一样。这样做的好处，自然是不言而喻的。也比较符合常规的美学思想。

然后是状态机的问题，尽量不要写出太大的状态机，宁愿用一些小型的状态机来相互关联。（除非你希望不被老板替换掉而写出很炫的，不过那样你自己以后也会很麻烦，因为你的记性并不像你想像中那么可靠）工具会耗尽心机去实现你一个很宏伟的状态机，结果可能还不能让人足够满意。

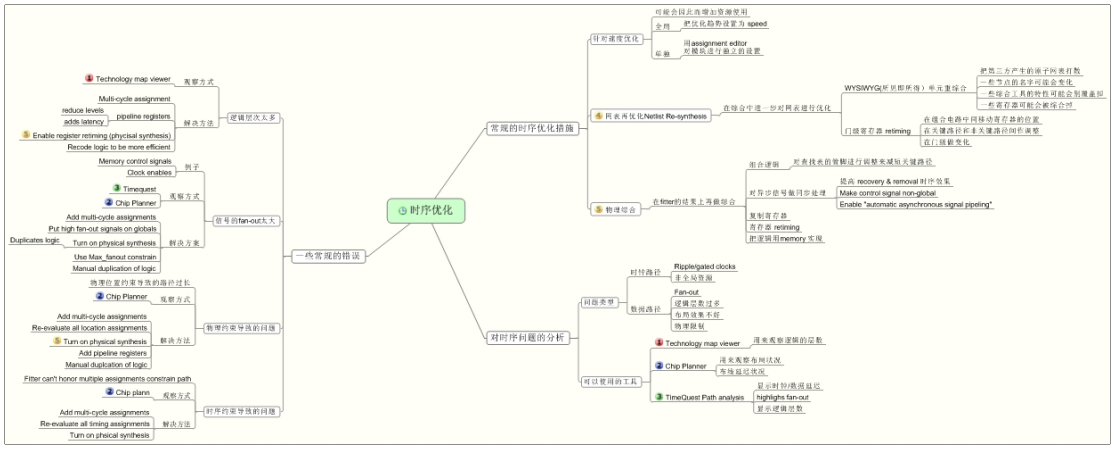
Glue logic. 就是两个模块之间的黏合逻辑。这或许是很难避免的。但他们往往会成为优化的一个盲点。这种三不管地区，最容易导致脏乱差等社会问题。所以，尽量把它们放进某个模块的势力范围。

在知道了写代码的大概规则以后，我们来看看约束。约束本身并不是优化，而是给优化制定的一个目标。你需要工具达到什么样的目标，你当然尽可能的要告诉工具，否则让他这么放任自流下去，后果会很严重。所以最简单的，你必须要提供所有的时序信息。好比，所有的时钟频率，所有的管脚时序要求。这样 fitter 才会有一个着眼点，针对距离目标最远的路径去努力。时序报告也才有可能报告哪些路径是没有通过要求的。我还是推荐大家使用 timequest 来做时序约束，好处是，它可能对你的时序约束和你的设计对照做分析，在做时序分析之前，先对你的约束做分析，然后告诉你，你有多少该做的事情而没有做的（为被约束的路径）还有多少你要求做的，而没有被做的（被忽略的时序要求）。

这里提出一个话题，false path，有人叫假路，有人叫错路，都是英语惹得祸，我们还是叫它flase path。具体来说，就是不该存在的路，或者说，即使存在，也没人去走的路。那么对这种路，我们是可以让工具看都不用看的。关于 false path，其实倒不用太过着急。大可以在发生问题以后，再去看是不是。否则让你把每条 false path 挑出来，其实也是个蛮无聊的事情。

对时钟的约束，要重点关注两个现象。首先是尽量少的在时钟路径上引入逻辑。这样认为的造成了时钟和时钟之间的 skew。我们都知道这不是什么好事情。另外就是一种上下沿都需要用来采集数据的时钟。对于时钟的约束有很多的地方需要注意，否则你的电路都不知道会飞到哪里去。不过这东西需要一些体验，靠嘴巴说是说不出感觉的。

现在来说约束中最重要的一个关键，不要过约束。过约束的坏处一大堆，增加编译时间（你可能不太在乎），资源使用过度（可能也还可以忍受），导致其他的时序问题（那可是个大麻烦了）如果你对自己的约束有些不太放心，又或者说可能器件和器件之间会有很细微的差别，你可以给约束做一些余量，但是过约束是万万要不得的。



- **时序优化:**

写了半天，终于开始写优化了，首先我们看看这个时序优化的问题。还是先看几个关键词吧。

优化目标：一个是全局的设置，告诉工具你这个设计的优化偏向。当时序优化偏重要的时候，可以设置为 Speed。当然这使以牺牲 area, power 为代价的。也可以对某个独立模块做局部设置，就是说在这个模块中，优化目标是时序。

网表再综合(netlist re-synthesis): QuarutsII 支持一些第三方的综合工具（说实话，Quaruts 自己的综合工具，也还是可以的）。他们把代码变成可以映射到 FPGA 上的网表文件，就是由一堆查找表和寄存器组成的东西。再综合会把这些网表还原成为与非门结构的电路，然后再根据 quartus 综合工具的算法再做综合。这当然未必是一件好事，可能会因此而破坏了原来工具中一些好的算法做出来的逻辑而导致结果更差。

物理综合：这是时序工具中效果最明显的工具。它对 fitter 以后的结果，在关键路径上，对电路进行调整。既然布局布线都做好了，还有什么可以做的呢？有很多。首先是组合电路。组合电路，无非就是一堆查找表的连接。而布局布线本身是一个比较随机的过程。在完成以后，我们会发现一些路径是比较差的，那么我们可以对这些路径做一些调整，他们选择更短的一条路径，同时功能保持不变。对异步控制信号做一级同步流水线。这样可以规避一些 recovery 和 removal 的时序问题。还有一个功能是复制寄存器，比如说一个寄存器的 fanout 比较多，或者说，其中的一条线会连到比较远的地方。那么我们可以在那个比较远的地方复制一个和这个寄存器完全一样的寄存器，这样可以大大提高时序效果。当然这需要牺牲一个寄存器了。还有一个好玩的事情，是寄存器的 re-timing，如果我们把寄存器看成流水线中的一层的话，它其实就是位于一条路径中的某个点而已。他的前后两个组合电路的延迟可能是不同的。而可能恰巧那个长的变成了一个关键路径（就是大大的坏的一条），那么我们是不是可以移动一下这个寄存器的位置，让大家的时间可以平衡一下，而结果是时序上去了。但是在功能上会有一点小小的变化，所以这个选项还是慎用。

可能造成的几种时序问题：

时钟路径导致问题，这个时钟可能是个 gated clock,或者非全局时钟。这个调整的余地相对比较小。

数据路径导致问题，这个比较复杂，首先可能是 Fan-out 太大导致的延迟过大。也有可能是逻辑门的层数过多导致。或者布局布线的随机随得不好，两个点的距离很大。还有就是一些物理限制，比如 DSP block 之间的相对位置。

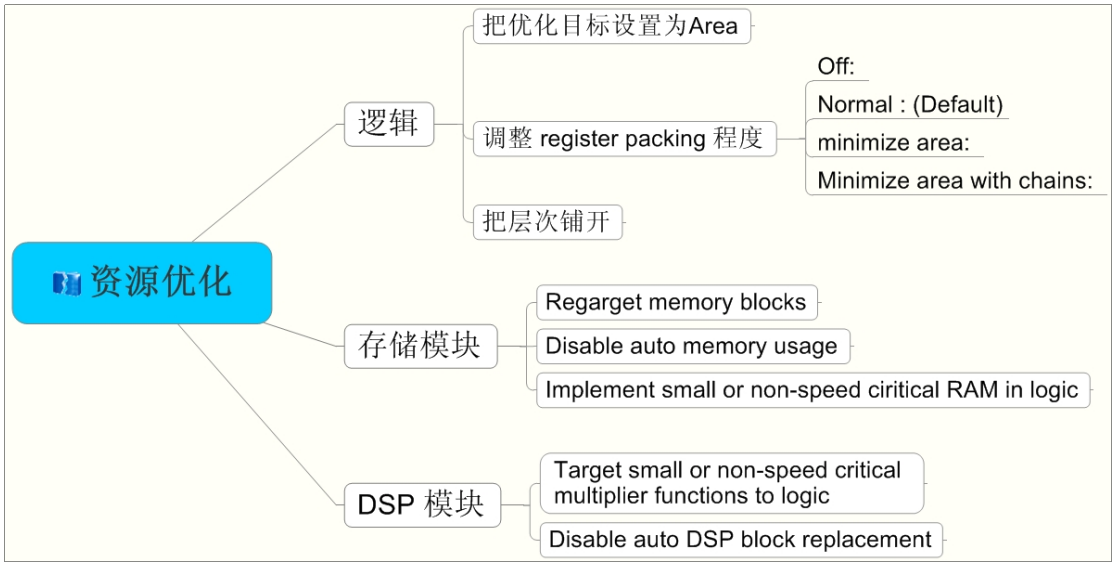
用来观察时序问题的工具：

Technology map viewer: 这东西，实在是没办法用中文说了，我私自反了，恐怕你会恨死我，因为在 Quartus 里面死活找不到了。它是可以用来观察你的综合结果以后的电路实现的。所以用它，你可以看到你的电路经过了多少层次的路径。

Chip planner: 这是用来看你的电路具体在芯片上的实现的，通过它你可以观察到相对之间的位置的距离，和连线上的延迟。

TimeQuest: 这是时序分析的最佳工具，无论是观察时钟或者路径延迟，fanout, 或者逻辑层次，都可以使用它。虽然不一定比其他的工具更直观，但是它是最全面的。

针对一些经典的可能性，表格里面列出了一些，大家可以自己看一下。

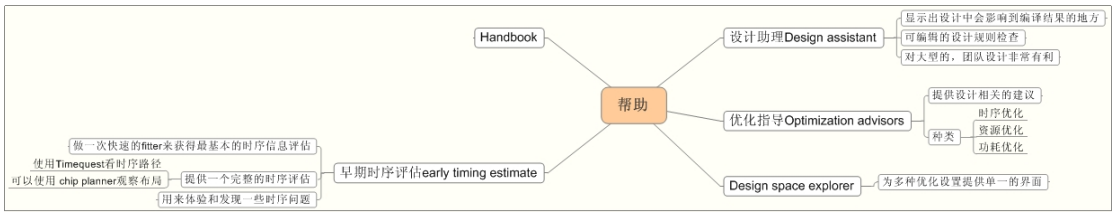


- **资源优化**

和时序优化相对的，我们可以给 Quartus 一个优化指导方向，当你把它设置为 area 的时候，工具会尽量考虑资源的使用。可以做全局设置，也可以做局部设置。

Register packing: 这是一个针对 LE 资源优化的工具，在新的 Quartus 中一共有七级设置。比如把两个互不相干的电路挤到同一个 LE 中。根据等级设置的不同，会有不同的资源消耗结果，当然付出的代价就是时序的降低。

当然对特定模块的使用，是需要一个整体规划的。什么样的状况下去使用存储器资源，什么样的状况下使用 DSP-block，都需要有一定的评估。更多的时候，其实在写代码的时候就已经做好了计划。



- **帮手：**

相信大家已经有点晕了，幸运的时候，我们不是一个人在战斗。我们还是有很多帮手的。让我们来看看这些家伙：

Design assistant： 这是一个会提供给你一些提示的助手工具。他会显示出设计中会导致编译结果的地方。你可以自己定制一些设计规则，他会帮助你去监督你自己，尤其在一个团队设计中，对大家设计的同一性有比较好的效果。

Optimization advisor: 优化指导，会提出一些相关的优化选项的指导。当然这种指导是非常通用的，不可能非常针对你自己的应用，所以在是否使用这些建议的时候，还是要自己权衡一下。所以简单的了解一些优化选项，还是有必要的。否则，莫名其妙的就被工具耍了。

Early timing estimate: 这个工具会做一些最基本的布局布线，这样它可以很快的得到一些时序方面的信息，这在你做时序约束，和logic lock设置的时候有很多的指导意义，同时也节省了一些时间。

Seeds: 谁都靠不上，我们靠老天。布局布线基本上是一个随机的事情，没有人能够了解结果能使什么样子的。而seeds就好像一个赌博用的骰子一样，告诉编译从什么状况开始布局。没有人知道seeds 中的具体某个数字代表什么样的状况，所以就是不断的碰运气吧。给大家一个迷信，似乎17这个数字特别好，结果会比较理想一些。

Design Space Explorer: 这个是一个实在没办法的办法，但凡还有一线生路，最好不要用这个东西。他会自动的对你的设计根据一些设置，不断的尝试布局布线。比如使用不同的优化方法，不同的算法，甚至不同的种子来做编译，最后给你一个最符合约束要求的编译结果。但是这样的工作量有多大，大家可想而知了。如果你有比较多的时间，你可以试试把它扔在那里跑个几天。

Handbook: 啥也不用说了。