关于 QuartusII 的一些事情

QuartusII 其实就是一个转换器。一个把你理解的逻辑语言转换成为器件能理解的语言，然后可以让 FPGA 按照你的想法去工作。我们写的那些 VHDL 啦，Verilog 啦什么的，其实都是人类自己定义的语言，对机器来说，就是对牛弹琴了，它没可能知道人类这些傻瓜坐在那里想做什么。所以为了交流，我们需要让他们理解我们的意图，而你也不至于因此而去学牛说话，所以，我们需要用 QuartusII，因为 QuartusII 就是帮助你进行这种翻译的工具。

我们分两部分来完成这么一件事情，首先把你的逻辑思路转变成用已经有的元件搭建出来的电路。好比你说我要一个加法，器件没你那么聪敏，他不知道什么叫做加法，加法对他来说是没有意义的。工具会把加法转换成为一组逻辑，用与，或，与非，或非这些乱七八糟的连在一起，变成和加法结果一样的电路。这样器件一看就知道了，哦，对了，我有这些的。当然这个时候你再去看那些东西，可能就蒙了。这个过程就是综合。综合结果是一个网表文件(netlist)，也就是一堆很无聊的电路。而这种电路还只是停留在概念上，并没有映射到实物上面。

然后工具会做第二件事情: fitting。QuartusII 把你选择的器件找出来，对照它拥有的资源来放刚才转成的电路。FPGA 里面的资源都是现成已经做好的，好处是你不需要做，坏处是，你也改不了。第一步首先是放置，就是把那些逻辑一个个放到器件的相应位置上。最后，把所有的放置好的点连接起来。这样，你的思想就在硬件上面完美的体现出来了。然后我们再回过来看看每个点都在做什么，把这些信息存成一个文件，以后你只需要每次告诉器件这个文件，就可以实现你的设计。

这一切都不需要你来做，工具都可以自动完成，因为对器件世界的理解，工具比你要熟悉的多，所以你可以相信，他一定比你做的好。但是这样是不是太容易了呢？当然不会，否则就不好玩了，你需要知道工具去实现，因为很多事情是工具不晓得的，你需要告诉工具你的要求来实现你的设计。这就是约束。好像女孩子喜欢漂亮，当然不会随便穿戴，要点缀这里，束缚那里，让自己看上去至少很美。打扮没有约束，可能顶多就是难看点。但是翻译过程没有约束，就完全不是一个东西了。所以，约束很重要，后果很严重。你要告诉工具很多信息。比如：时钟信息，工具不可能聪明到知道你从外面送进来的时钟是多少的。比如管脚信息，你要告诉他哪些输入，哪些是输出，而这些输入输出分别应该是什么样子的。这些是最基本的信息，还可以提出一些更苛刻的要求出来。比如说功耗。QuartusII 在完成编译以后会对编译的结果进行分析，看是不是能满足你的要求。如果不满足，它可以再试几次。最后告诉你：帅哥，按照你的要求，我们完成了工作了，或者说，小子，你的要求是在太过分了，我做不了。那么可能你得调整一下设计，代码，约束条件，甚至器件。

到此为止，其实 QuartusII 的工作已经可以说完成了。但如果只是这样，你一定会很郁闷，因为你并不知道你的设计是不是真的可以工作。或者说，你会想知道你的设计为什么不工作。所以必须提供仿真和 debug 工具。QuartusII 提供一个相对比较简单的仿真工具，让你可以通过画波形图的方式来做仿真。这种工具非常容易用，但是能力比较有限。他可以很直白的实现激励和观察输出波形的平台，但是他无法实现很好的交互能力。好比你需要因为一个信号

而发动下一个动作时？你恐怕很难猜得出来什么时候这个信号就来了吧。所以很多时候，我们会比较喜欢用第三方工具 ModelSim. 这个东西不好的地方就是你需要自己写仿真代码。你会发现这是一件蛮痛苦的事情，很多时候仿真代码比电路代码本身还要复杂。但是，真的没有办法，我很同情你，因为只有你才知道你想做什么，没有人能代替你来写仿真。

仿真毕竟只是仿真而已，不是真的，很多时候你会发现电路还是不能工作，虽然仿真看上去完全没问题。千万不要太自信的以为这一定就是工具的 bug，从而很兴高采烈的报告说，我找到你们一个 bug。绝大多数的情况下，要么是因为你的约束有问题，要么是因为你的仿真和真实情况不符合。所以，QuartusII 不得不提供一种途径，让你看到，芯片里面到底发生了一些什么事情，那就是在片 debug 工具，signaltapII。我知道名字不太好听，但是其实真的还蛮有用的。

QuartusII 就是这样一个工具，很简单，但要玩得好，还是需要一点功夫的。后面我会比较细化的聊聊怎么玩这个东西。我不会很老土的说怎么创建一个工具，然后按这里，按那里。这些要你自己去玩，去摸索，否则，完全没有乐趣了。