關於 QuartusII 的一些事情

QuartusII 其實就是一個轉換器。一個把你理解的邏輯語言轉換成爲器件能理解的語言，然後可以讓 FPGA 按照你的想法去工作。我們寫的那些 VHDL 啦，Verilog 啦什麼的，其實都是人類自己定義的語言，對機器來說，就是對牛彈琴了，它沒可能知道人類這些傻瓜坐在那裏想做什麼。所以爲了交流，我們需要讓他們理解我們的意圖，而你也不至於因此而去學牛說話，所以，我們需要用 QuartusII，因爲 QuartusII 就是幫助你進行這種翻譯的工具。

我們分兩部分來完成這麼一件事情，首先把你的邏輯思路轉變成用已經有的元件搭建出來的電路。好比你說我要一個加法，器件沒你那麼聰敏，他不知道什麼叫做加法，加法對他來說是沒有意義的。工具會把加法轉換成爲一組邏輯，用與，或，與非，或非這些亂七八糟的連在一起，變成和加法結果一樣的電路。這樣器件一看就知道了，哦，對了，我有這些的。當然這個時候你再去看那些東西，可能就蒙了。這個過程就是綜合。綜合結果是一個網表檔(netlist)，也就是一堆很無聊的電路。而這種電路還只是停留在概念上，並沒有映射到實物上面。

然後工具會做第二件事情: fitting。QuartusII 把你選擇的器件找出來，對照它擁有的資源來放剛才轉成的電路。FPGA 裏面的資源都是現成已經做好的，好處是你不需要做，壞處是，你也改不了。第一步首先是放置，就是把那些邏輯一個個放到器件的相應位置上。最後，把所有的放置好的點連接起來。這樣，你的思想就在硬體上面完美的體現出來了。然後我們再回過來看看每個點都在做什麼，把這些資訊存成一個檔，以後你只需要每次告訴器件這個檔，就可以實現你的設計。

這一切都不需要你來做，工具都可以自動完成，因爲對器件世界的理解，工具比你要熟悉的多，所以你可以相信，他一定比你做的好。但是這樣是不是太容易了呢？當然不會，否則就不好玩了，你需要知道工具去實現，因爲很多事情是工具不曉得的，你需要告訴工具你的要求來實現你的設計。這就是約束。好像女孩子喜歡漂亮，當然不會隨便穿戴，要點綴這裏，束縛那裏，讓自己看上去至少很美。打扮沒有約束，可能頂多就是難看點。但是翻譯過程沒有約束，就完全不是一個東西了。所以，約束很重要，後果很嚴重。你要告訴工具很多資訊。比如：時鐘資訊，工具不可能聰明到知道你從外面送進來的時鐘是多少的。比如管腳資訊，你要告訴他哪些輸入，哪些是輸出，而這些輸入輸出分別應該是什麼樣子的。這些是最基本的資訊，還可以提出一些更苛刻的要求出來。比如說功耗。QuartusII 在完成編譯以後會對編譯的結果進行分析，看是不是能滿足你的要求。如果不滿足，它可以再試幾次。最後告訴你：帥哥，按照你的要求，我們完成了工作了，或者說，小子，你的要求是在太過分了，我做不了。那麼可能你得調整一下設計，代碼，約束條件，甚至器件。

到此爲止，其實 QuartusII 的工作已經可以說完成了。但如果只是這樣，你一定會很鬱悶，因爲你並不知道你的設計是不是真的可以工作。或者說，你會想知道你的設計爲什麼不工作。所以必須提供仿真和 debug 工具。QuartusII 提供一個相對比較簡單的仿真工具，讓你可以通過畫波形圖的方式來做仿真。這種工具非常容易用，但是能力比較有限。他可以很直白的實

現激勵和觀察輸出波形的平臺，但是他無法實現很好的交互能力。好比你需要因爲一個信號而發動下一個動作時？你恐怕很難猜得出來什麼時候這個信號就來了吧。所以很多時候，我們會比較喜歡用第三方工具 ModelSim. 這個東西不好的地方就是你需要自己寫仿真代碼。你會發現這是一件蠻痛苦的事情，很多時候仿真代碼比電路代碼本身還要複雜。但是，真的沒有辦法，我很同情你，因爲只有你才知道你想做什麼，沒有人能代替你來寫仿真。

仿真畢竟只是仿真而已，不是真的，很多時候你會發現電路還是不能工作，雖然仿真看上去完全沒問題。千萬不要太自信的以爲這一定就是工具的 bug，從而很興高采烈的報告說，我找到你們一個 bug。絕大多數的情況下，要麼是因爲你的約束有問題，要麼是因爲你的仿真和真實情況不符合。所以，QuartusII 不得不提供一種途徑，讓你看到，晶片裏面到底發生了一些什麼事情，那就是在片 debug 工具，signaltapII。我知道名字不太好聽，但是其實真的還蠻有用的。

QuartusII 就是這樣一個工具，很簡單，但要玩得好，還是需要一點功夫的。後面我會比較細化的聊聊怎麼玩這個東西。我不會很老土的說怎麼創建一個工具，然後按這裏，按那裏。這些要你自己去玩，去摸索，否則，完全沒有樂趣了。