你的 Q-zone，你做不了主

我的地盤我做主，這其實是一句鬼話。你很少真的有什麼地盤你可以做主的，因爲你很難作爲規則製造者存在。你只有更好的依循規則，你才能更好的讓事情按照你的想法去做。所以爲了做主你的地盤，你最好依照一些規則，而不是按照自己的喜好來做，好比寫代碼。

上電初始值

在通常的狀況下，所有的門在上電的時候輸出爲低。但是這並不是不能改變的。你可以把上電設置爲高，這樣綜合工具可能會做兩種事情，把輸出反向，或者使用 preset 控制（如果存在的話）把初始值放進門裏。

當時上電爲高的做法，並不是非常必要，因爲你其實是可以使用重定信號來獲得你想要的初始狀態的。

如果你覺得這是必須的，那麼有幾種方法你可以做：

- 首先是在 QuartusII 裏面你可以針對某個或者某些 門設置 power-up level 爲高或低。
- 在代碼中使用 altera_attribute
- 直接寫代碼設置初始值：

```
reg   q = 1'b1;
always @ (posedge clk or posedge aclr)

begin
      if (aclr)
                  q <= 1'b0;
      else
                  q <= d;
end
```

門的次級管理信號

每個門都有一些次級的管理裝置，好比清除信號啊，時鐘使能信號啊。而這些裝置當然都有他們自己的操作規律。如果你在寫代碼的時候可能適當的使用它們，那麼綜合的時候很容易就可以使得王八看到綠豆，大家都對上了。其實實現一個功能是沒有問題的，但是如果你把功能按照它的自然規律來實現，從資源消耗還是很划算的。當然我知道大家現在都很有錢，不太在乎這些的，但是省吃儉用似乎還是硬體設計師德傳統美德。你會發現年資越大的工程師在這方面越是注意，所以，如果你希望在別人眼裏看上去比較牛的話，適當的使用這種手段，還是蠻炫的。

我們把這些信號按照優先順序排列一下（不知道什麼是優先順序？找本字典先）

1. 非同步清零信號 – aclr
2. 上電重定信號, - pre
3. 非同步載入信號 – aload
4. 使能信號 – ena
5. 同步清零信號 – sclr

6. 同步載入信號 – sload
7. 資料登錄信號 – data

我建議大家可以嘗試自己用這些信號來做一個門出來看看，嘗試怎麼可以使用這些信號。然後使用 quartus 來編譯驗證自己的結果。這是一種非常有趣的玩法。首先，它可以使你對器件的結構更加瞭解，同時也鍛煉了你寫代碼的能力。一旦兩相結合，就可以牛的亂七八糟的。

雙向信號

首先說明一件基本知識，在 FPGA 設計中，只有在輸入輸出上可以使用雙向信號，雙向信號是不能使用在內部邏輯上的。一定不要用這種信號，否則工具會綜合出一個你都不知道會是什麼東西的東西。

針對一個雙向埠，你需要把它變成一個輸入信號 in，一個輸出信號：out, 和一個輸出使能信號: output_enable. 所以代碼其實很簡單：

```
Assign birsignal = output_enable ? out: 1'bz;
Assign in = birsignal
```

這裏有一個小小的提示，在寫代碼的時候突然不太清楚語法怎麼寫的時候，你可以在 quartus 裏面按一下右鍵，你可以發現一個 insert template…的選擇。試試看吧。

加法器

加法器的做法就比較有趣了，這會涉及到器件本身的結構問題。相比大家都看到了，FPGA 的單元結構是前面一個查找表結果，後面接一個門（寄存器）。我們當然希望可能盡可能的使用這些結構。

最好的狀況是什麼？就是使用前面的那個查找表做一個加法，然後直接送到後面的那個門裏面，然後傳到下一級做加法。這樣說好像很沒意思，我們舉個例子來看看可以怎麼玩：

比如我們要做這樣一個加法 res = A + B + C + D + E

我們首先需要知道一下加法是怎麼做出來的，加法本身其實是很簡單的邏輯，但是問題是，你不是一個人在戰鬥，你需要有進位元，你還需要送進位元出去。但是你真的等到進位來了你才做計算，這樣一個 100 位的加法，你大概要等到天黑了。所以，其實在實現一個加法的時候，我們會同時做兩次，一次假設進位爲一，一次假設進位爲二。然後用前一級下來的進位來進行選擇。所以用這種方式我們就可以理解怎麼進一步在 FPGA 中實現加法了。

我們有兩種查找表類型

首先是 4 輸入查找表 （ Stratix, Cyclone, 和其他那些老古董）。

作爲四輸入的查找表，就比較適合兩個數加，然後可以直接連到後面的門上去關一下。進位元是通過進位元鏈鏈結的。 所以這樣我們可以做成這樣的演算法。

一號門 = A + B　　二號門 = C + D.

三號門 = 一號門 + 二號門

四號門 = 三號門 + E

這樣通過一個三層門的結構做一個五輸入的加法器，來達到最好的效果。
6 輸入查找表 （StratixII, CycloneII, 和其他那些比較新的器件）
在新的結構中，我們可以使用 6 輸入的查找表，這樣，就可以用三個數加在一起 變成：

一號門 = A + B + C
二號門 = 一號門 + D + E

這樣就變成一種兩重門的結構。

說這些，可以說是一種提醒吧。我們在寫代碼的時候就可以考慮硬體的結構，用這種方法，讓你的實現可以變得更加專業起來。在 FPGA 中的硬體結構都已經是固定的，所以如果按照你的地盤的規律來寫，那一定更完美。大家不妨嘗試各種寫代碼的方式來實現。

狀態機

狀態機是設計過程中的核心部分，所以我們需要特別的提一下寫狀態機的一些注意事項。爲了實現利益最大化，建議在 FPGA 中使用 one hot 模式的狀態機，而在 CPLD 中使用最少比特數的狀態機。在具體的設計中需要注意的是：

- 把狀態機寫全，也就是說不要漏寫了 Default：。 沒有這個首先會出現什麼？對了，會有假門(latch)。
- 狀態機作爲控制核心部分，儘量把它和演算法功能和資料分離開來。好像你看到好的流水線，控制流水線的電腦和流水線本身是分開的。這樣可以保持相對的獨立性。
- 如果一種操作設計到幾個狀態，儘量把操作剝離狀態機本身。
- 使用一個簡單的重定信號來定義上電狀態。如果你的狀態機會被比較多的重定信號重定的話，工具就不會把它當作狀態機來對待。

總之，儘量的保持狀態機的很傻很單純是很重要的。儘量的不要加重核心部分的複雜性。其實道理很簡單，好比在一個公司裏面，真正在工作的，其實一定不是一個這個公司的核心。