你的 Q-zone，你做不了主

我的地盘我做主，这其实是一句鬼话。你很少真的有什么地盘你可以做主的，因为你很难作为规则制造者存在。你只有更好的依循规则，你才能更好的让事情按照你的想法去做。所以为了做主你的地盘，你最好依照一些规则，而不是按照自己的喜好来做，好比写代码。

上电初始值

在通常的状况下，所有的门在上电的时候输出为低。但是这并不是不能改变的。你可以把上电设置为高，这样综合工具可能会做两种事情，把输出反向，或者使用 preset 控制（如果存在的话）把初始值放进门里。

当时上电为高的做法，并不是非常必要，因为你其实是可以使用复位信号来获得你想要的初始状态的。

如果你觉得这是必须的，那么有几种方法你可以做：

- 首先是在 QuartusII 里面你可以针对某个或者某些 门设置 power-up level 为高或低。
- 在代码中使用 altera_attribute
- 直接写代码设置初始值：

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

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

门的次级管理信号

每个门都有一些次级的管理装置，好比清除信号啊，时钟使能信号啊。而这些装置当然都有他们自己的操作规律。如果你在写代码的时候可能适当的使用它们，那么综合的时候很容易就可以使得王八看到绿豆，大家都对上了。其实实现一个功能是没有问题的，但是如果你把功能按照它的自然规律来实现，从资源消耗还是很划算的。当然我知道大家现在都很有钱，不太在乎这些的，但是省吃俭用似乎还是硬件设计师德传统美德。你会发现年资越大的工程师在这方面越是注意，所以，如果你希望在别人眼里看上去比较牛的话，适当的使用这种手段，还是蛮炫的。

我们把这些信号按照优先级排列一下（不知道什么是优先级？找本字典先）

1. 异步清零信号 – aclr
2. 上电复位信号, - pre
3. 异步载入信号 – aload
4. 使能信号 – ena
5. 同步清零信号 – sclr
6. 同步载入信号 – sload

7. 数据输入信号 – data

我建议大家可以尝试自己用这些信号来做一个门出来看看，尝试怎么可以使用这些信号。然后使用 quartus 来编译验证自己的结果。这是一种非常有趣的玩法。首先，它可以使你对器件的结构更加了解，同时也锻炼了你写代码的能力。一旦两相结合，就可以牛的乱七八糟的。

双向信号

首先说明一件基本知识，在 FPGA 设计中，只有在输入输出上可以使用双向信号，双向信号是不能使用在内部逻辑上的。一定不要用这种信号，否则工具会综合出一个你都不知道会是什么东西的东西。

针对一个双向端口，你需要把它变成一个输入信号 in，一个输出信号：out, 和一个输出使能信号: output_enable. 所以代码其实很简单：

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

这里有一个小小的提示，在写代码的时候突然不太清楚语法怎么写的时候，你可以在 quartus 里面按一下右键，你可以发现一个 insert template…的选择。试试看吧。

加法器

加法器的做法就比较有趣了，这会涉及到器件本身的结构问题。相比大家都看到了，FPGA 的单元结构是前面一个查找表结果，后面接一个门（寄存器）。我们当然希望可能尽可能的使用这些结构。

最好的状况是什么？就是使用前面的那个查找表做一个加法，然后直接送到后面的那个门里面，然后传到下一级做加法。这样说好像很没意思，我们举个例子来看看可以怎么玩：

比如我们要做这样一个加法 res = A + B + C + D + E

我们首先需要知道一下加法是怎么做出来的，加法本身其实是很简单的逻辑，但是问题是，你不是一个人在战斗，你需要有进位，你还需要送进位出去。但是你真的等到进位来了你才做计算，这样一个 100 位的加法，你大概要等到天黑了。所以，其实在实现一个加法的时候，我们会同时做两次，一次假设进位为一，一次假设进位为二。然后用前一级下来的进位来进行选择。所以用这种方式我们就可以理解怎么进一步在 FPGA 中实现加法了。

我们有两种查找表类型

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

作为四输入的查找表，就比较适合两个数加，然后可以直接连到后面的门上去关一下。进位是通过进位链链接的。 所以这样我们可以做成这样的算法。

一号门 = A + B　　二号门 = C + D.

三号门 = 一号门 + 二号门

四号门 = 三号门 + E

这样通过一个三层门的结构做一个五输入的加法器，来达到最好的效果。
6 输入查找表 （StratixII, CycloneII, 和其他那些比较新的器件）
在新的结构中，我们可以使用 6 输入的查找表，这样，就可以用三个数加在一起 变成：

一号门 = A + B + C
二号门 = 一号门 + D + E

这样就变成一种两重门的结构。

说这些，可以说是一种提醒吧。我们在写代码的时候就可以考虑硬件的结构，用这种方法，让你的实现可以变得更加专业起来。在 FPGA 中的硬件结构都已经是固定的，所以如果按照你的地盘的规律来写，那一定更完美。大家不妨尝试各种写代码的方式来实现。

状态机

状态机是设计过程中的核心部分，所以我们需要特别的提一下写状态机的一些注意事项。为了实现利益最大化，建议在 FPGA 中使用 one hot 模式的状态机，而在 CPLD 中使用最少比特数的状态机。在具体的设计中需要注意的是：

- 把状态机写全，也就是说不要漏写了 Default：。 没有这个首先会出现什么？对了，会有假门(latch)。
- 状态机作为控制核心部分，尽量把它和算法功能和数据分离开来。好像你看到好的流水线，控制流水线的电脑和流水线本身是分开的。这样可以保持相对的独立性。
- 如果一种操作设计到几个状态，尽量把操作剥离状态机本身。
- 使用一个简单的复位信号来定义上电状态。如果你的状态机会被比较多的复位信号复位的话，工具就不会把它当作状态机来对待。

总之，尽量的保持状态机的很傻很单纯是很重要的。尽量的不要加重核心部分的复杂性。其实道理很简单，好比在一个公司里面，真正在工作的，其实一定不是一个这个公司的核心。