我想用一种比较有效的方式来描述关于系统设计的一些事情。设计是没有一个固定规则的过程，否则也就没有乐趣可言了。虽然有很多标准，有很多限制，但是依然有非常多的变化空间。其实最重要的事情，是知道什么是重要的，什么是次要的。精力应该集中在重要的东西上面，而不是一些细节。做系统设计需要有一种大的气魄，一些不拘小节的气质，当然，还需要别出心裁的创意。FPGA 本身提供了一个非常灵活的平台，如何最大化的利用这个平台，是我们脱离呆板的嵌入式系统的一个重要关键，也是如何玩转系统的关键。

不求甚解之 NiosII

所有的系统都是由模块组成的，或大或小的模块，拼接成一个大积木。所以我们首先需要了解这些模块 (IP)。关于怎么去了解一个 IP，其实是很重要的问题。NiosII CPU 作为一个比较大型的模块，可以作为一个例子来讲。一个关键词是不求甚解。不求甚解的目的，并不是偷懒，而是更准确，更有针对性，更快捷。IP 的作用就是为了完成一个特定的功能，所以我们并不需要知道它是如何实现的，事实上，由于很多的 IP 都是加密的代码，所以也不可能知道具体的电路状态。同时也不需要花很多的时间把文档里面的每一行都了解清楚。作为工程师，大家的脾气一般都是一种超强的好奇心和钻研精神的集合。而往往会钻进牛角尖里面。而作为系统设计，是需要有一种粗旷型的大气魄，不需要在细节上浪费时间。你会发现很多的细节是没有意义的。并不是说我们不需要去研究细节，细节是很重要的，但是细节需要在被用到的时候才去关注就好了。

作为一个 IP，最重要的，其实是接口。因为你最重要的是需要知道是怎么让它工作起来，而不是它怎么工作的。所以在看文档的时候，最主要看的就是接口信号，对所有的信号的作用有一个了解。NiosII 使用的是 Avalon MM 点对点接口，这其实是一个非常有趣的接口，因为它的交流更加短平快一些。它与普通的 PCI 接口不同的地方是，他可以支持同时多线控制。因为它没有总线的概念，不会在总线被占据的时候，其他任何通讯都无法进行。NiosII 是在 SOPC builder 中被直接使用的，我们不需要知道具体有哪些信号，因为没有非常需要，我们是看不到这些接口的。在 NiosII 中，我们有两个 Master Avalon MM 接口，一个是 Instruction Master Port, 这是 CPU 用来读取指令的接口。CPU 通过这个端口从 Memory 上读取指令。另一个是 Data master port, 很简单，这是用来连接数据通道的。比如说你要读取的数据，你要存储的数据，都是走这个通道。这两个端口可以连接同一个内存，在这种时候需要特别小心，很有可能自己把自己的指令给改掉了。但是反过来思考一下，其实我们可以做什么？可以按照状况改变软件代码。NiosII 中还有第三个端口，这是用来做 Debug 用的端口。还有其他的一些接口，比如 TCM 接口。我们需要知道这些接口的存在，但是不需要知道细节，只有在用到的时候再去看相关的文档就好了。

第二个需要关注的问题就是参数设置。这里面是有讲究的。IP 是厂家做出来的通用模块，不是为你而特制的，所以必然有一些是你不需要的方面。我们可以通过参数的修改，让它尽量的接近我们的需求。有很多人在做设计的时候是有思维定势的，而且这种定势的顽固性很强。这很容易对环境产生一种叛逆思想。就是说除了他自己假想出来的做法，其他的一切都

是不对的，或者说不好的。而这在使用 IP 的时候，会遭遇到意想不到的痛苦的。所以，尽量不要依靠假设来臆想了模块的设置。而是尽量的适应环境，来配置自己的设计。作为一个 FPGA 的玩家，这种依照环境来改变的能力是必须的。

NiosII 的参数中首先是指令集或者说 CPU 复杂度的选择，有三个选择，根据不同的选择，CPU 的能力会有不同，当然使用的资源也是完全不一样的

- NiosII/E (经济型)能力最弱，当然资源也最小(600-700 LE )
- NiosII/S (中间型) 中庸配置，资源消耗是 1200-1400 LE
- NiosII/F (快速型) 能力最强配置，资源消耗是 1400-1800LE

然后我们考虑 Cashes 的设置，Cash 有两种，一种是用来做指令缓存的，一种是用来做数据缓存的。Cash 的大小对程序的运行速度是有影响的。当然也没必要使用过多的资源。够用就好了。

再了解一下 Jtag Debug 的模式选择，一共有五个等级的选择。和选择 CPU 一样，从简单到复杂。一般来说选择 Level2 也就够用的。

自定义指令设置。这是最有价值的设置。别忘记了，我们是在 FPGA 的世界里。所以 CPU 并不像在其他地方那样的铁板一块。我们可以选择使用自定义的指令。所谓自定义指令，并不是一个软件宏或者函数。而是一块硬件。当 CPU 调用到这个指令的时候，事实上它调用的就是这个硬件模块，它被嵌入在 CPU 中。而这其实就是 NiosII 好玩的地方。

好了，现在我们要考虑的问题，就是使用。使用 CPU 的方法，当然就是软件编程。NiosII 的软件其实是非常简单的。就是普通的 C，或者 C++，需要做的就是不断的对端口的地址读啊，写啊，计算数据，就好了。但也可能非常的复杂，因为你不仅需要了解软件编程，你更需要了解你使用的那些硬件，那些外设模块。NiosII 的编程很硬件的依赖性是很强的。针对比较大型的一些外设，可以写一些 HAL 程序。这是类似于驱动的一些指令。而软件只需要调用这些 API 就可以了。大部分的 NiosII 程序是不需要使用操作系统的，作为一个嵌入式系统的控制核心，更多的是一些存储式的读写，算法的计算的操作。除非你需要运行一些网络协议啊，什么的。但其实我们可以用更加解构的方式来看待一个操作系统。操作系统其实就是给我们提供了一大堆的操作指令而已。没什么更特别的作用了。所以，思考一下吧。

我用了不求甚解的方式来介绍了一下 NiosII。这是一种偷懒的方式。我要做的，其实就是把一些关键的点指出来，大家可以去看，同时不把时间消耗在细枝末节上面。要把 NiosII 完整的说清楚，当然不是这么三言两语的就可以的了，否则事情也太简单了些了。我希望可以对大家有所帮助的地方就是，对一件复杂的事情，找一个聪明的方式。