我想用一種比較有效的方式來描述關於系統設計的一些事情。設計是沒有一個固定規則的過程，否則也就沒有樂趣可言了。雖然有很多標準，有很多限制，但是依然有非常多的變化空間。其實最重要的事情，是知道什麼是重要的，什麼是次要的。精力應該集中在重要的東西上面，而不是一些細節。做系統設計需要有一種大的氣魄，一些不拘小節的氣質，當然，還需要別出心裁的創意。FPGA 本身提供了一個非常靈活的平臺，如何最大化的利用這個平臺，是我們脫離呆板的嵌入式系統的一個重要關鍵，也是如何玩轉系統的關鍵。

不求甚解之 NiosII

所有的系統都是由模組組成的，或大或小的模組，拼接成一個大積木。所以我們首先需要瞭解這些模組 (IP)。關於怎麼去瞭解一個 IP，其實是很重要的問題。NiosII CPU 作爲一個比較大型的模組，可以作爲一個例子來講。一個關鍵字是不求甚解。不求甚解的目的，並不是偷懶，而是更準確，更有針對性，更快捷。IP 的作用就是爲了完成一個特定的功能，所以我們並不需要知道它是如何實現的，事實上，由於很多的 IP 都是加密的代碼，所以也不可能知道具體的電路狀態。同時也不需要花很多的時間把文檔裏面的每一行都瞭解清楚。作爲工程師，大家的脾氣一般都是一種超強的好奇心和鑽研精神的集合。而往往會鑽進牛角尖裏面。而作爲系統設計，是需要有一種粗曠型的大氣魄，不需要在細節上浪費時間。你會發現很多的細節是沒有意義的。並不是說我們不需要去研究細節，細節是很重要的，但是細節需要在被用到的時候才去關注就好了。

作爲一個 IP，最重要的，其實是介面。因爲你最重要的是需要知道是怎麼讓它工作起來，而不是它怎麼工作的。所以在看文檔的時候，最主要看的就是介面信號，對所有的信號的作用有一個瞭解。NiosII 使用的是 Avalon MM 點對點介面，這其實是一個非常有趣的介面，因爲它的交流更加短平快一些。它與普通的 PCI 介面不同的地方是，他可以支持同時多線控制。因爲它沒有匯流排的概念，不會在匯流排被佔據的時候，其他任何通訊都無法進行。NiosII 是在 SOPC builder 中被直接使用的，我們不需要知道具體有哪些信號，因爲沒有非常需要，我們是看不到這些介面的。在 NiosII 中，我們有兩個 Master Avalon MM 介面，一個是 Instruction Master Port, 這是 CPU 用來讀取指令的介面。CPU 通過這個埠從 Memory 上讀取指令。另一個是 Data master port, 很簡單，這是用來連接資料通道的。比如說你要讀取的資料，你要存儲的資料，都是走這個通道。這兩個埠可以連接同一個記憶體，在這種時候需要特別小心，很有可能自己把自己的指令給改掉了。但是反過來思考一下，其實我們可以做什麼？可以按照狀況改變軟體代碼。NiosII 中還有第三個埠，這是用來做 Debug 用的埠。還有其他的一些介面，比如 TCM 介面。我們需要知道這些介面的存在，但是不需要知道細節，只有在用到的時候再去看相關的文檔就好了。

第二個需要關注的問題就是參數設置。這裏面是有講究的。IP 是廠家做出來的通用模組，不是爲你而特製的，所以必然有一些是你不需要的方面。我們可以通過參數的修改，讓它儘量的接近我們的需求。有很多人在做設計的時候是有思維定勢的，而且這種定勢的頑固性很強。這很容易對環境產生一種叛逆思想。就是說除了他自己假想出來的做法，其他的一切都是不對的，或者說不好的。而這在使用 IP 的時候，會遭遇到意想不到的痛苦的。所以，儘

量不要依靠假設來臆想了模組的設置。而是儘量的適應環境，來配置自己的設計。作爲一個 FPGA 的玩家，這種依照環境來改變的能力是必須的。

NiosII 的參數中首先是指令集或者說 CPU 複雜度的選擇，有三個選擇，根據不同的選擇，CPU 的能力會有不同，當然使用的資源也是完全不一樣的

- NiosII/E (經濟型)能力最弱，當然資源也最小(600-700 LE )
- NiosII/S (中間型) 中庸配置，資源消耗是 1200-1400 LE
- NiosII/F (快速型) 能力最強配置，資源消耗是 1400-1800LE

然後我們考慮 Cashes 的設置，Cash 有兩種，一種是用來做指令緩存的，一種是用來做資料緩存的。Cash 的大小對程式的運行速度是有影響的。當然也沒必要使用過多的資源。夠用就好了。

再瞭解一下 Jtag Debug 的模式選擇，一共有五個等級的選擇。和選擇 CPU 一樣，從簡單到複雜。一般來說選擇 Level2 也就夠用的。

自定義指令設置。這是最有價值的設置。別忘記了，我們是在 FPGA 的世界裏。所以 CPU 並不像在其他地方那樣的鐵板一塊。我們可以選擇使用自定義的指令。所謂自定義指令，並不是一個軟體巨集或者函數。而是一塊硬體。當 CPU 調用到這個指令的時候，事實上它調用的就是這個硬體模組，它被嵌入在 CPU 中。而這其實就是 NiosII 好玩的地方。

好了，現在我們要考慮的問題，就是使用。使用 CPU 的方法，當然就是軟體編程。NiosII 的軟體其實是非常簡單的。就是普通的 C，或者 C++，需要做的就是不斷的對埠的位址讀啊，寫啊，計算資料，就好了。但也可能非常的複雜，因爲你不僅需要瞭解軟體編程，你更需要瞭解你使用的那些硬體，那些外設模組。NiosII 的編程很硬體的依賴性是很強的。針對比較大型的一些外設，可以寫一些 HAL 程式。這是類似於驅動的一些指令。而軟體只需要調用這些 API 就可以了。大部分的 NiosII 程式是不需要使用作業系統的，作爲一個嵌入式系統的控制核心，更多的是一些存儲式的讀寫，演算法的計算的操作。除非你需要運行一些網路協定啊，什麼的。但其實我們可以用更加解構的方式來看待一個作業系統。作業系統其實就是給我們提供了一大堆的操作指令而已。沒什麼更特別的作用了。所以，思考一下吧。

我用了不求甚解的方式來介紹了一下 NiosII。這是一種偷懶的方式。我要做的，其實就是把一些關鍵的點指出來，大家可以去看，同時不把時間消耗在細枝末節上面。要把 NiosII 完整的說清楚，當然不是這麼三言兩語的就可以的了，否則事情也太簡單了些了。我希望可以對大家有所幫助的地方就是，對一件複雜的事情，找一個聰明的方式。