FPGA 学习的一些误区汇总整理

1、不熟悉 FPGA 的内部结构，不了解可编程逻辑器件的基本原理。

FPGA 为什么是可以编程的?恐怕很多菜鸟不知道，他们也不想知道。因为他们觉得这是无关紧要的。他们潜意识的认为可编程嘛，肯定就是像写软件一样啦。软件编程的思想根深蒂固，看到 Verilog 或者 VHDL 就像看到 C 语言或者其它软件编程语言一样。一条条的读，一条条的分析。 如果这些菜鸟们始终拒绝去了解为什么 FPGA 是可以编程的，不去了解 FPGA 的内部结构，要想学会 FPGA 恐怕是天方夜谭。虽然现在 EDA 软件已经非常先进，像写软件那样照猫画虎也能综合出点东西，但也许只有天知道 EDA 软件最后综合出来的到底是什么。

也许点个灯，跑个马还行。这样就是为什么很多菜鸟学了 N 久以后依然是一个菜鸟的原因。那么 FPGA 为什么是可以“编程”的呢?首先来了解一下什么叫“程”。其实“程”只不过是一堆具有一定含义的 01 编码而已。编程，其实就是编写这些 01 编码。

只不过我们现在有了很多开发工具，通常都不是直接编写这些 01 编码，而是以高级语言的形式来编写，最后由开发工具转换为这种 01 编码而已。对于软件编程而言，处理器会有一个专门的译码电路逐条把这些 01 编码翻译为各种控制信号，然后控制其内部的电路完成一个个的运算或者是其它操作。所以软件是一条一条的读，因为软件的操作是一步一步完成的。

而 FPGA 的可编程，本质也是依靠这些 01 编码实现其功能的改变，但不同的是 FPGA 之所以可以完成不同的功能，不是依靠像软件那样将 01 编码翻译出来再去控制一个运算电路，FPGA 里面没有这些东西。FPGA 内部主要三块：可编程的逻辑单元、可编程的连线和可编程的 IO 模块。

可编程的逻辑单元是什么?其基本结构某种存储器(SRAM、FLASH 等)制成的 4 输入或 6 输入 1 输出地“真值表”加上一个 D 触发器构成。任何一个 4 输入 1 输出组合逻辑电路，都有一张对应的“真值表”，同样的如果用这么一个存储器制成的 4 输入 1 输出地“真值表”，只需要修改其“真值表”内部值就可以等效出任意 4 输入 1 输出的组合逻辑。这些“真值表”内部值是什么?就是那些 01 编码而已。如果要实现时序逻辑电路怎么办?

这不又 D 触发器嘛，任何的时序逻辑都可以转换为组合逻辑+D 触发器来完成。但这毕竟只实现了 4 输入 1 输出的逻辑电路而已，通常逻辑电路的规模那是相当的大哦。那怎么办呢?这个时候就需要用到可编程连线了。在这些连线上有很多用存储器控制的链接点，通过改写对应存储器的值就可以确定哪些线是连上的而哪些线是断开的。

这就可以把很多可编程逻辑单元组合起来形成大型的逻辑电路。最后就是可编程的 IO，这其实是 FPGA 作为芯片级使用必须要注意的。任何芯片都必然有输入引脚和输出引脚。有可编程的 IO 可以任意的定义某个非专用引脚(FPGA 中有

专门的非用户可使用的测试、下载用引脚)为输入还是输出，还可以对 IO 的电平标准进行设置。总归一句话，FPGA 之所以可编程是因为可以通过特殊的 01 代码制作成一张张“真值表”，并将这些“真值表”组合起来以实现大规模的逻辑功能。

不了解 FPGA 内部结构，就不能明白最终代码如何变到 FPGA 里面去的。也就无法深入的了解如何能够充分运用 FPGA。现在的 FPGA，不单单是有前面讲的那三块，还有很多专用的硬件功能单元，如何利用好这些单元实现复杂的逻辑电路设计，是从菜鸟迈向高手的路上必须要克服的障碍。而这一切，还是必须先从了解 FPGA 内部逻辑及其工作原理做起。

2、错误理解 HDL 语言，怎么看都看不出硬件结构。

HDL 语言的英语全称是：HardwareDescriptionLanguage，注意这个单词 Description,而不是 Design。老外为什么要用 Description 这个词而不是 Design 呢?因为 HDL 确实不是用用来设计硬件的，而仅仅是用来描述硬件的。描述这个词精确地反映了 HDL 语言的本质，HDL 语言不过是已知硬件电路的文本表现形式而已，只是将以后的电路用文本的形式描述出来而已。而在编写语言之前，硬件电路应该已经被设计出来了。

语言只不过是将这种设计转化为文字表达形式而已。但是很多人就不理解了，既然硬件都已经被设计出来了，直接拿去制作部就完了，为什么还要转化为文字表达形式再通过 EDA 工具这些麻烦的流程呢?其实这就是很多菜鸟没有了解设计的抽象层次的问题，任何设计包括什么服装、机械、广告设计都有一个抽象层次的问题。就拿广告设计来说吧，最初的设计也许就是一个概念，设计出这个概念也是就是一个点子而已，离最终拍成广告还差得很远。

硬件设计也是有不同的抽象层次，每一个层次都需要设计。最高的抽象层次为算法级、然后依次是体系结构级、寄存器传输级、门级、物理版图级。使用 HDL 的好处在于我们已经设计好了一个寄存器传输级的电路，那么用 HDL 描述以后转化为文本的形式，剩下的向更低层次的转换就可以让 EDA 工具去做了，这就大大的降低了工作量。

这就是可综合的概念，也就是说在对这一抽象层次上硬件单元进行描述可以被 EDA 工具理解并转化为底层的门级电路或其他结构的电路。在 FPGA 设计中，就是在将这以抽象层级的意见描述成 HDL 语言，就可以通过 FPGA 开发软件转化为问题 1 中所述的 FPGA 内部逻辑功能实现形式。

HDL 也可以描述更高的抽象层级如算法级或者是体系结构级，但目前受限于 EDA 软件的发展，EDA 软件还无法理解这么高的抽象层次，所以 HDL 描述这样抽象层级是无法被转化为较低的抽象层级的，这也就是所谓的不可综合。所以在阅读或编写 HDL 语言，尤其是可综合的 HDL，不应该看到的是语言本身，而是要看到语言背后所对应的硬件电路结构。如果看到的 HDL 始终是一条条的代码，那么这种人永远摆脱不了菜鸟的宿命。假如哪一天看到的代码不再是一行行的代码而是一块一块的硬件模块，那么恭喜脱离了菜鸟的级别，进入不那么菜的鸟级别。

3、FPGA 本身不算什么，一切皆在 FPGA 之外这一点恐怕也是很多学 FPGA 的菜鸟最难理解的地方。

FPGA 是给谁用的?很多学校解释为给学微电子专业或者集成电路设计专业的学生用的，其实这不过是很多学校受资金限制，卖不起专业的集成电路设计工具而用 FPGA 工具替代而已。其实 FPGA 是给设计电子系统的工程师使用的。这些工程师通常是使用已有的芯片搭配在一起完成一个电子设备，如基站、机顶盒、视频监控设备等。

当现有芯片无法满足系统的需求时，就需要用 FPGA 来快速的定义一个能用的芯片。前面说了，FPGA 里面无法就是一些“真值表”、触发器、各种连线以及一些硬件资源，电子系统工程师使用 FPGA 进行设计时无非就是考虑如何将这些以后资源组合起来实现一定的逻辑功能而已，而不必像 IC 设计工程师那样一直要关注到最后芯片是不是能够被制造出来。本质上和利用现有芯片组合成不同的电子系统没有区别，只是需要关注更底层的资源而已。

要想把 FPGA 用起来还是简单的，因为无非就是那些资源，在理解了前面两点再搞个实验板，跑跑实验，做点简单的东西是可以的。而真正要把 FPGA 用好，那光懂点 FPGA 知识就远远不够了。因为最终要让 FPGA 里面的资源如何组合，实现何种功能才能满足系统的需要，那就需要懂得更多更广泛的知识。

目前 FPGA 的应用主要是三个方向：第一个方向，也是传统方向主要用于通信设备的高速接口电路设计，这一方向主要是用 FPGA 处理高速接口的协议，并完成高速的数据收发和交换。这类应用通常要求采用具备高速收发接口的 FPGA，同时要求设计者懂得高速接口电路设计和高速数字电路板级设计，具备 EMCEMI 设计知识，以及较好的模拟电路基础，需要解决在高速收发过程中产生的信号完整性问题。FPGA 最初以及到目前最广的应用就是在通信领域，一方面通信领域需要高速的通信协议处理方式，另一方面通信协议随时在修改，非常不适合做成专门的芯片。因此能够灵活改变功能的 FPGA 就成为首选。

到目前为止 FPGA 的一半以上的应用也是在通信行业。第二个方向，可以称为数字信号处理方向或者数学计算方向，因为很大程度上这一方向已经大大超出了信号处理的范畴。例如早就在 2006 年就听说老美将 FPGA 用于金融数据分析，后来又见到有将 FPGA 用于医学数据分析的案例。在这一方向要求 FPGA 设计者有一定的数学功底，能够理解并改进较为复杂的数学算法，并利用 FPGA 内部的各种资源使之能够变为实际的运算电路。目前真正投入实用的还是在通信领域的无线信号处理、信道编解码以及图像信号处理等领域，其它领域的研究正在开展中，之所以没有大量实用的主要原因还是因为学金融的、学医学的不了解这玩意。

不过最近发现欧美有很多电子工程、计算机类的博士转入到金融行业，开展金融信号处理，相信随着转入的人增加，FPGA 在其它领域的数学计算功能会更好的发挥出来，而我也有意做一些这些方面的研究。不过国内学金融的、学医的恐怕连数学都很少用到，就不用说用 FPGA 来帮助他们完成数学_运算了，这个问题只有再议了。第三个方向就是所谓的 SOPC 方向，其实严格意义上来说这个已经在 FPGA 设计的范畴之内，只不过是利用 FPGA 这个平台搭建的一个嵌入式系统

的底层硬件环境，然后设计者主要是在上面进行嵌入式软件开发而已。设计对于FPGA本身的设计时相当少的。但如果涉及到需要在FPGA做专门的算法加速，实际上需要用到第二个方向的知识，而如果需要设计专用的接口电路则需要用到第一个方向的知识。

就目前SOPC方向发展其实远不如第一和第二个方向，其主要原因是因为SOPC以FPGA为主，或者是在FPGA内部的资源实现一个“软”的处理器，或者是在FPGA内部嵌入一个处理器核。但大多数的嵌入式设计却是以软件为核心，以现有的硬件发展情况来看，多数情况下的接口都已经标准化，并不需要那么大的FPGA逻辑资源去设计太过复杂的接口。而且就目前看来SOPC相关的开发工具还非常的不完善，以ARM为代表的各类嵌入式处理器开发工具却早已深入人心，大多数以ARM为核心的SOC芯片提供了大多数标准的接口，大量成系列的单片机嵌入式处理器提供了相关行业所需要的硬件加速电路，需要专门定制硬件场合确实很少。通常是在一些特种行业才会在这方面有非常迫切的需求。即使目前Xilinx将ARM的硬核加入到FPGA里面，相信目前的情况不会有太大改观，不要忘了很多老掉牙的8位单片机还在嵌入式领域混呢，嵌入式主要不是靠硬件的差异而更多的是靠软件的差异来体现价值的。

笔者曾经看好的是cypress的Psoc这一想法。和SOPC系列不同，Psoc的思想史载SOC芯片里面去嵌入那么一小块FPGA，那这样其实可以满足嵌入式的那些微小的硬件接口差异，比如某个运用需要4个USB，而通常的处理器不会提供那么多，就可以用这么一块FPGA来提供多的USB接口。而另一种运用需要6个UART，也可以用同样的方法完成。对于嵌入式设计公司来说他们只需要备货一种芯片，就可以满足这些设计中各种微小的差异变化。其主要的差异化仍然是通过软件来完成。但目前cypress过于封闭，如果其采用ARM作为处理器内核，借助其完整的工具链。同时开放IP合作，让大量的第三方为它提供IP设计，其实是很有希望的。但目前cypress的日子怕不太好过，Psoc的思想也不知道何时能够发光。

4、数字逻辑知识是根本

无论是FPGA的哪个方向，都离不开数字逻辑知识的支撑。FPGA说白了是一种实现数字逻辑的方式而已。如果连最基本的数字逻辑的知识都有问题，学习FPGA的愿望只是空中楼阁而已。而这，恰恰是很多菜鸟最不愿意去面对的问题。数字逻辑是任何电子电气类专业的专业基础知识，也是必须要学好的一门课。很多人无非是学习了，考个试，完了。如果不能将数字逻辑知识烂熟于心，养成良好的设计习惯，学FPGA到最后仍然是雾里看花水中望月，始终是一场空的。以上四条只是我目前总结菜鸟们在学习FPGA时所最容易跑偏的地方，FPGA的学习其实就像学习围棋一样，学会如何在棋盘上落子很容易，成为一位高手却是难上加难。要真成为李昌镐那样的神一般的选手，除了靠刻苦专研，恐怕还确实得要一点天赋。