【博客藏经阁丛书】

FPGA深度解析

樊继明 陆锦宏 编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

藏经阁丛书

FPGA深度解析

樊继明 陆锦宏 编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

内 容 简 介

本书是一本 FPGA 开发经验总结式的书籍，以实例讲解的方式详细介绍了 FPGA 的概念、使用场景及开发流程，对 FPGA 的芯片架构做了详细说明；同时，对 FPGA 的开发流程，包括可综合 RTL 代码的编写及验证、工具的综合及布局布线、静态时序分析等概念做了详细分析。在此基础上，还详细介绍了 FPGA 常用处理模块的设计，对重要的基础性设计模块，例如异步 FIFO、高速 SerDes 接口以及高速 LVDS 的接收、抽取滤波器的设计等也进行了深入讲解。

本书的内容全面、实用，讲解通俗易懂，适合没有形成 FPGA 设计思想概念但是有一定开发基础的设计人员参考。

图书在版编目(CIP)数据

FPGA 深度解析 / 樊继明，陆锦宏编著. -- 北京 :
北京航空航天大学出版社，2015.5
ISBN 978-7-5124-1759-5

Ⅰ. ①F… Ⅱ. ①樊…②陆… Ⅲ. ①可编程序逻辑器件—系统开发 Ⅳ. ①TP332.1

中国版本图书馆 CIP 数据核字(2015)第 078589 号

FPGA 深度解析

樊继明 陆锦宏 编著

责任编辑 董立娟 张耀军

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@buaacm.com.cn 邮购电话：(010)82316936

涿州市新华印刷有限公司印装 各地书店经销

*

开本：710×1 000 1/16 印张：16.75 字数：377 千字

2015 年 5 月第 1 版 2015 年 5 月第 1 次印刷 印数：3 000 册

ISBN 978-7-5124-1759-5 定价：39.00 元

前言

本书的由来

从学生时代开始接触 FPGA 再到参加工作，笔者一直在与 FPGA/ASIC 的逻辑设计打交道。刚开始接触 FPGA 的时候，脑海里还一直受软件式编程语言的影响，没有形成对 Verilog HDL 硬件描述语言、硬件设计的本质理解。参加工作从事大型的逻辑设计后，才逐渐理解到逻辑设计的本质。这么多年来，笔者阅读过很多关于 Verilog HDL 逻辑设计的书籍或者资料，大多数侧重于阐述 HDL 语言的语法本身，而很少描述该语言硬件设计本质，于是萌生了一个自己整理的念头，这也是笔者写这本书的一个初衷。

本书的特点

作为一名 FPGA 逻辑设计工程师，笔者认为能够对自己写出的代码综合后的电路有个想象总是件好事，因此，在本书中，特别是后面的设计例子，笔者并没有给出实际的代码。这其实隐含了对逻辑设计的一个看法，即在编写 RTL 代码前，一定要对自己设计的逻辑电路、数据处理流程、控制流程有一个最清晰、直观的理解。并把其记录下来或者文档化，那么后续的代码编写是一件相当轻松的事情。这其实也是大公司提倡从需求、系统方案到概要设计、详细设计然后才是编码的流程的目的，即把事情想清楚再动手。这也是本书提倡的一种理念，也是笔者多年开发过程中受益的一个经验总结。

本书并不是一本大而全的 FPGA 入门书籍，确切说，本书是一本经验总结式书籍，但是这个经验总结并不是发散的，而是章节安排有严密的逻辑关系，从 FPGA 的使用场景、设计流程，再到重要基础部件，慢慢过渡到典型设计，因此也可以说是一个从架构到具体设计由浅入深的一个经验总结。

尽管对基于 FPGA 的逻辑设计、仿真验证并不像 ASIC 设计那样要求全面，但是在 FPGA 逻辑设计规模越来越大的今天，一个设计编译的时间有可能达到几个小时，这就决定了此时的 FPGA 逻辑设计不能像写一个单片机软件那样，一遇到 Bug 就重新修改某段代码、重新编译再烧写到芯片里看结果，此时就要求对设计做好充分的仿真验证，即最大程度地保证功能的正常验证通过，这样才能编译并下载到 FPGA 里，否则效率实在太低。

基于 FPGA 的逻辑设计实际上是在设计数字逻辑电路，既然是电路，那么就有一定的物理限制，因此 FPGA 工程师不能只局限于编写 RTL 代码，同时也要对设计的时

序分析、优化有一定的认识，这也是笔者花较多篇幅介绍静态时序分析章节的原因。

本书还对 FPGA 设计的常用模块设计理念进行了详细的分析，比如说异步 FIFO，这是一个非常重要的单元模块，如果对其原理不够了解，使用的时候就有可能出问题，因此本书对异步 FIFO 的原理以及使用注意事项做了深入阐述。此外，高速 SerDes 接口设计、抽取滤波器、FFT 的设计等，都是工程师有可能遇到的。另外，高速 LVDS 信号的接收，特别是源同步的设计，需要一定的约束技巧或者是对 I/O 要有一定的理解，初学者容易出问题，因此这一章也是笔者着重阐述的一章。

主要内容

本书是一本 FPGA 开发经验总结式的书籍，以实例讲解的方式详细介绍了 FPGA 的概念、使用场景及开发流程，对 FPGA 的芯片架构做了详细说明；同时，对 FPGA 的开发流程，包括可综合 RTL 代码的编写及验证、工具的综合及布局布线、静态时序分析等概念做了详细分析。在此基础上，还详细介绍了 FPGA 常用处理模块的设计，对重要的基础性设计模块，例如异步 FIFO、高速 SerDes 接口以及高速 LVDS 的接收、抽取滤波器的设计等也进行了深入讲解。

本书的内容全面、实用，讲解通俗易懂，适合没有形成 FPGA 设计思想概念但是有一定开发基础的设计人员参考。

感谢北京航空航天大学出版社的支持，是他们的坚持才能让本书与读者见面。感谢在工作中给予笔者指导和启发的领导和同事，特别是中兴通讯的曾献君教授、系统架构师周海涛，以及理邦精密仪器的陈敏、刘旺锋以及逻辑设计组的成员。

限于自身的水平与经验，出错在所难免，因此希望读者能够见谅，对本书有任何见解，欢迎与笔者联系：logic_thinker@qq.com。

樊继明　陆锦宏
2015 年 3 月于深圳

目 录

第1章 FPGA简介

1.1 什么是FPGA

1.1.1 FPGA简述

FPGA(Field Programble Gate Array)是一种完成通用功能的可编程逻辑芯片,即可以对其进行编程实现某种逻辑处理功能。它集成了大量的原始逻辑资源(触发器、查找表LUT和布线),并且提供了可配置的I/O口及硬IP(Block RAM、PLL、DSP、Serdes等),依赖于工程师采用HDL(Hardware Description Language,硬件描述语言)语言进行编码,各逻辑并行工作来实现指定的功能。它是基于硬件描述的芯片。

通俗来说,FPGA就像一块面包板。还有一种可编程逻辑芯片CPLD(Complex Programmable Logic Device,复杂可编程逻辑器件),其与FPGA一样具有"面包板"的属性。但是,FPGA与CPLD又有区别:FPGA的逻辑规模比CPLD大得多,拥有更丰富的触发器资源、DSP单元以及支持更加丰富的I/O数量与类型。简单地说,FPGA是一块规模极大、复杂度极高的"面包板",而CPLD则是一块规模很小、复杂度低得多的"面包板"。

FPGA芯片上有许多门电路器件和少量的I/O、IP(专用电路),以及许多待连接的走线。没进行编码之前,这些电路只有本身的基本功能,并不具备系统级的功能。编码之后,这块面包板上门和门之间被连接起来,I/O、IP按照设计进行配置和连接,最后形成一块连接了的、具有特定功能的电路板。就像一些与非门和触发器在面包板上按照累加器的功能进行连接,并用来点亮七段数码管一样,只不过很多基于FPGA设计的逻辑电路规模非常大,大到难以手工连接这块"面包板",所以需要通过编写HDL代码从而借助IDE自动完成这些工作。

工程师在编写完代码后,在集成开发环境(IDE,Integrated Design Environment)中经过综合(Synthesis)、映射(Map)、布局布线(PnR, Place and Route),生成可配置二进制文件。可配置二进制文件指定了FPGA内部布线节点的配置(互连)信息,以及各I/O、硬IP等的配置信息。

1.1.2 FPGA 与 MCU 芯片的区别

FPGA 在外观上与一般芯片并无异样，本质上也如此。实际上，FPGA 就是一种芯片，但与 ARM、51 单片机、PowerPC……相比，FPGA 又是一种完全不同的芯片。

若采用基于 51 的 MCU 进行设计，则首先会用 C 语言进行编码。编码之前 MCU 本身就只有一个 51 内核以及其他外设，没有编码就没有功能，芯片只能是芯片本身。例如，要对粮仓温度进行监测，当监测温度超过临界值时，则给出报警信号，并将该信号驱动到警报喇叭上，以此提醒工作人员对粮仓温度过高的地方进行处理，降低温度避免损失。此时，硬件工程师就需要将一个温度传感器连接到 51 单片机的 ADC 上（假设该 MCU 集成了 ADC，如果没有则须外接），并且将驱动警报喇叭的信号通过片上 DAC 输出至功放并驱动警报喇叭。软件工程师则需要编写代码，使 MCU 每隔一段时间对 ADC 上的数值进行读取，判断是否超过临界值。假如超过临界值，则将一段数字化的报警信号通过 DAC 转化成模拟信号，经过功放驱动警报喇叭，这样就完成了项目需求里规定的功能。

显然，假如不针对该项目的功能需求进行分析和编码，即使连接了硬件，预定的功能也无法实现。MCU 是一种以共性存在的器件，提供了能以某一指令集进行编码的处理器内核，并且集成了若干外部设备。用户可以用它来接收或者驱动外部器件，并通过编码来指定它们的行为。该编码确定了在指定条件下 MCU 的指令执行顺序。

而对于 FPGA 来说，通常不会像一般的 MCU 嵌入 ADC 或者 DAC 这样的外设。因此，假如要实现上述项目的功能，需要将一个温度传感器连接到外部 ADC，然后将 ADC 的数据接口连接到 FPGA（I^2C、SPI 或者其他）。然后将 FPGA 的报警音频数字接口连接至外部 DAC 上，并通过运放驱动警报喇叭。工程师同样需要编码。与 51 不同，FPGA 最终并非通过指定条件下 MCU 的指令执行顺序来确保功能，而是以同时并行的方式工作。工程师采用硬件描述语言进行编码，即可以实时读取 ADC 接口上的数据再将数据与临界值进行比较，并且决定是否报警。

此外，与一般的 MCU 不同，这块“面包板”上的所有电路总是同时工作。在每个时钟有效沿，所有的触发器总是在触发，由输入（诸如使能这样的控制信号）决定是否翻转，就像手表里的所有机械器件一样，同时在运转，“滴答滴答”地驱动着指针转动。这点不像 MCU，MCU 的软件代码是逐条指令顺序（跳转）执行的。

1.2 FPGA 的应用场景

1. 新型 SoC FPGA 的出现及其应用场景

FPGA 有特定的应用场景，不是任何项目需求都适合用 FPGA 实现。FPGA 的应用是基于芯片的逻辑资源或硬件设计专用电路。换而言之，基于 FPGA 的应用方案一旦设计完成，其实现的功能就完全确定。要在 FPGA 上实现完全不同的功能或者应

用,则意味着进行完全不同的设计:完全不同的PCB与引脚分配,完全不同的RTL代码,或者完全不同的芯片规格……除了重新规划设计方案没有太好的选择。

不同的是,软件工程师在面临这样的问题时,只要内存和Flash足够,就几乎可以随心所欲地修改或者增减应用程序。软件工程师几乎不必担心为了实现一个新的应用程序而去换一个MCU,或者更换一个新的操作系统,或者牺牲其他应用程序的功能……

但在实际应用中,工程师可以在FPGA上搭建或者集成MCU,即所谓的SoC(System on Chip)。搭建SoC的好处是:

① 减少线路板(PCB)上的芯片数量,减少芯片面积,降低板级走线(互连)的数量和复杂度。原先可能需要使用一个或更多的MCU,采用SoC可以在FPGA上实现若干核的MCU,使它们协同工作。这些MCU以FPGA为载体,也就是说它们在物理上只共享FPGA的芯片面积。将MCU集成在FPGA内部,大量的板级走线变成了FPGA内部布线,只剩下必要的外部设备接口与FPGA连接,大大降低了板级走线的数量和复杂度。

② 使PCB更容易设计并且更可靠。由于降低了板级走线的数量和复杂度,绘制原理图和进行PCB布线时都会变得更加容易。并且,减少连线也意味着减少出错的节点,使得电路更加可靠。

③ 降低功耗。将若干MCU集成到FPGA里,可以更加有效、更加统一地进行电源管理。芯片的内核电压通常与工艺与制程有关。一般来讲,更先进的工艺往往意味着更低的内核电压,而FPGA的制程通常是比较先进的。将若干MCU集成到FPGA的好处是明显的,也就是将不同MCU的不同内核电压统一到FPGA的内核电压上,因而能够降低内核功耗。此外,SoC使用更少的I/O,同样也降低了I/O的功耗。

④ 减少电磁干扰(EMI),提高系统的电磁兼容性(EMC)。

⑤ 实现更灵活、更复杂的功能。由于MCU之间的连接和组合是定制的,因此可以根据实际需求进行方案设计,实现比使用现有分立的MCU更加灵活、复杂的功能。

⑥ 提高数据吞吐量和带宽。FPGA可以提供实时且并行的高速串行数据接收,即多Lane Serdes的应用,可以解决外部高速大量数据到片内的传输问题;此外,FPGA可以根据实际需求的数据吞吐量和带宽来决定数据处理的并行度,实现一般CPU无法达到的高带宽、超高带宽能力。

⑦ 兼顾软件开发的灵活性与FPGA的专用性。SoC方案在集成了MCU之后,既可以实现MCU的功能,也能实现专用电路功能,使两者得到兼顾。

⑧ 成本。使用SoC与降低成本这项并非总是符合预期。在使用不同等级的FPGA时,成本会有差异,并且涉及MCU IPcore的授权费用。此外,FPGA作为特殊芯片,价格比ASIC和成熟大批量制造的MCU更高。因此,FPGA的SoC方案较少出现在消费类电子产品上,在专业领域里,比如通信处理器、工控、医疗电子、军事设备等应用广泛。

然而,SoC只是FPGA应用领域里的一部分,更多的应用场景则是基于独立MCU

与独立 FPGA 的设计方案，将灵活多变的功能与需求交给 MCU 与软件处理，将专用、特殊甚至是诸如接口转换这样的功能交给 FPGA 实现。

2. 传统 FPGA 的应用模式

选择使用 FPGA 有很多因素，最重要的一点是所要实现的功能采用常规的 MCU 实现并不容易，甚至是实现不了。MCU 不容易或者实现不了的功能，通常集中出现在大规模的算法和运算、高速数据和大吞吐量、系统实时反应、I/O 互连和转换等场合。

在一些领域，比如说示波器的设计开发中，示波器的采样率通常数倍于带宽，即使像 200M 带宽这样常见的示波器，其采样率也往往达到 G 级别；此外还须考虑样点位宽，这样就面临着 10G 级别的 I/O 带宽和数据吞吐量。

在数据处理方面，设计示波器面临的挑战在于：

- 从前端到芯片的数据传输通常是高速甚至是超高速的；
- 对大带宽甚至是超大带宽的数据进行存储；
- 数据的实时处理通常需要非常大的吞吐量；
- 实现 DSP 算法的代价与开销通常较高(例如 FFT 和 IFFT)。

常见的 MCU 难以实现这样实时的高速率数据传输：设想 10 Gbps 的数据带宽，在 32 位处理器中，即使数据读/写能够在单周期内完成，并且总线效率达到 100%，其总线频率仍然超过 300 MHz。在这种情况下，假如要优先保证采样数据的完整性，处理器的总线带宽就会被占死，处理器几乎无法再做其他工作了。即使是专用的 DSP 芯片在这方面也捉襟见肘。但在示波器领域，FPGA 却能应用得很好。FPGA 丰富的片上 LVDS 接口和专用 Serdes 收发器能轻松地适应这种需求和挑战，解决高速、大带宽的数据传输问题。FPGA 能提供几十对 LVDS，传输带宽为 10 Gbps 的数据仍显绰绰有余。在获取数据后，FPGA 仍然能通过 DDR 控制器将数据存储至 DDR3 中。

在信号处理方面，FPGA 丰富的片上资源可以用来设计高性能的 DSP 算法，并且只要在允许的范围内，工程师就可以根据实际所需的数据吞吐量来决定电路的并行度或者流水线，实现实时高速大吞吐量的数据运算。

总而言之，在很多领域里，有许多设计上的困难是 MCU 无法解决、而 FPGA 却能胜任的，因此 FPGA 能在这些领域中广泛应用。

1.3 FPGA 现状

目前较常见的 FPGA 产商有 Xilinx、Altera、Actel、Lattice。这 4 家公司里，Xilinx 和 Altera 能提供业界最先进的 FPGA 芯片。它们的高端芯片所采用的工艺甚至用上了世界领先的工艺与制程。中低端的产品线采用的工艺相对不那么先进——通常落后 1～2 代。

Lattice 的 FPGA 性能不如 Xilinx 和 Altera 强大，开发环境以及工具集比较不友善，偶尔会遇到奇奇怪怪的问题。与 Xilinx 和 Altera 相比，Lattice 能提供极具性价比

的FPGA。通常来讲，Lattice高一规格的芯片价格与Xilinx、Altera低一规格的芯片价格相当。正是这种错位竞争让Lattice在市场占有一席之地。

Actel与其他的FPGA产商不同，提供了独有的基于Flash技术的FPGA，而不是基于SRAM。由于FPGA内部采用Flash保存芯片的配置信息，因此无须像基于SRAM的芯片那样外挂E^2PROM或者Flash来存储配置文件。这样的好处是显而易见的，譬如FPGA具有更快的启动速度、更高的环境耐性和可靠度、更高的保密性等。但基于Flash技术的FPGA无法将规模做得比SRAM更大，这跟器件特性、工艺水平密切相关。Actel凭借其特有的Flash型FPGA，同样也在FPGA市场上占有一定的份额。

目前，业界提供了各种层次的FPGA。最低端的FPGA价格便宜，适合用在简单逻辑功能以及普通但琐碎的接口转换或者互连方案中。中端的FPGA通常具有较大规模的片上逻辑和专用的硬IP(DSP单元、PCI硬核、SRAM、DDR控制器等)，并且支持更多的I/O类型，芯片价格相对适中。高端的FPGA具有非常大的电路规模，通常会集成相当数量的高速I/O以满足业界对超高速、超高带宽的需求。此外，高端的FPGA越来越往SoC发展，提供单芯片系统解决方案，其价格昂贵，适合用在大型、复杂的设计方案中。

1.4 开发FPGA需要的HDL语言

FPGA逻辑设计语言采用HDL(Hardware Description Language)，即硬件描述语言。该语言其实也是一种数字芯片设计的语言，芯片功能通常是使用HDL进行行为描述然后经过综合和布局布线完成实际功能。这跟软件开发通常使用C或者Java进行编码然后编译有异曲同工之妙。

Verilog和VHDL是目前两大常见的HDL，标准分别为IEEE1364和IEEE1076，但HDL并不仅限于这两种(还有ABEL)。

一直以来，使用HDL进行FPGA开发是一种传统，但是在最新的技术发展中，特别是一些算法实现领域也开始逐渐出现新的设计方式。在很多情况下，FPGA可以用来实现复杂的DSP算法。这需要前期对算法进行建模，并分析算法的性能是否满足需求。然而采用HDL进行建模并分析算法性能显然是不可能的。一般情况下，C或者Matlab是很好的选择。所以在最新的技术中，有将Matlab代码或者OpenCL代码直接转化成RTL(可综合硬件电路)的工具。由C/OpenCL、Matlab向RTL转化，也就是直接从前期算法建模和性能分析的工作中向硬件电路转化，跨过了HDL编码阶段，极大地改变了原有的工作流程。这种颠覆性的改变在于它具有极大的便利性，直接跨越了RTL编码的阶段，显然可以节省大量的设计时间并缩短项目周期。

但在现阶段，该技术既有优势也有劣势。工具能将原型代码按照希望的结果进行转化，得到一个功能符合预期的RTL代码。但在目前，这样的美好愿望并不能很好地实现。由于工具并不知道设计人员希望以怎样的架构实现硬件电路，就像盖房子时不考虑房屋内部结构和居住的舒适程度一样，依靠“高级”工具得到的电路并不尽人意，其

实现的逻辑电路在效率、资源、功耗方面不如有经验的工程师通过 HDL 代码编写的逻辑电路。所以在技术尚未变革之前，无论是初学者还是有经验的工程师，使用 HDL 进行编码还是 FPGA 应用的主要手段。

谈到 HDL 编码就不得不提及代码规范的必要性。在行业中，编码规范总是会被较为规范的公司所重视。虽然公司不会严格要求每个工程师编写的代码具有完全一致的范式，但还是会给出一套指导性规范，要求工程师在这些规范范围内编写代码。按照编码规范进行编码，不同的工程师也能编写出风格一致或者相近的代码，从而使协同开发顺利进行，从而提高协同开发效率。

除此之外，编码规范通常要求编写详细的文件头和代码注释。文件头有助于描述模块功能并且留有模块作者的联系方式，有助于代码移交或者项目人员更替时其他成员能更快地了解该模块。

代码注释的好处是显而易见的。详细、完整的代码注释不仅能帮助其他成员在阅读、移植代码时更方便地理解设计细节；还能帮助自己在经过一段时间之后，因为修改或者移植的需求再阅读自己的代码时，能更快、更准确无误地回忆当初的设计思路与实现方法，更快地进行修改工作。

因此，在刚接触 FPGA 并使用 HDL 进行编码时，养成良好的编码规范对将来的职业生涯是很有帮助的。

1.5 FPGA 设计流程

FPGA 的开发流程，包括需求分析、方案制定、详细设计、编码、仿真、时序约束、综合、布局布线、后仿真、时序分析、生成下载文件、下载到 FPGA 进行调试等步骤以及步骤间的关系，如图 1-1 所示。

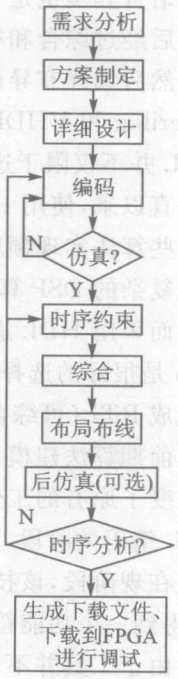

图 1-1 FPGA 开发流程

如图 1-1 所示，需求分析、方案制定、详细设计等是任何一个正规项目的必须阶段，其他步骤则是 FPGA 自有的设计流程。编码：用 HDL 语言进行编码。仿真：对 HDL 编码后的模块在计算机上进行功能仿真，确保功能正确；该仿真只是逻辑行为正确性的仿真，是一种不包含布局布线后的真实包含 FPGA 内部逻辑单元及逻辑单元走线延时的仿真。约束：对逻辑设计的时钟、引脚分配等进行描述，确保设计能够满足时钟和 I/O 的正确性。综合：把基于 HDL 语言的设计映射成基于 FPGA 内部逻辑单元的电路。布局布线：对综合后的电路在 FPGA 内部进行连线。后仿真：对布局布线后的结果进行计算机仿真，其与功能仿真类似，不同之处是后仿真加入了器件和连线延时，因此其结果更准确，但是耗时更长，一般设计很少有该流程。时序

分析：对布局布线后的结果进行分析，查看最终设计是否满足时钟频率要求。生成下载文件：把最终的设计结果变成反映FPGA里最终逻辑单元和连线的目标文件，最终下载到FPGA里并调试。图1-1反映的是一种比较常见的关系，并不是一个固定死的流程，例如可以把时序约束这一步骤放在仿真之前等。详细说明在后续章节中将会一一介绍，这里强调一下遵循FPGA设计流程的必要性。

良好的设计流程和完善的工作步骤显然能提高工作效率并且保证设计的质量。这里举两个不按照标准设计流程带来麻烦的例子。例子1：在项目开发过程中，假如涉及DSP算法，如果不对算法性能进行验证或者不对算法定点化进行分析，而是直接进入到后面的工作中，那么，即使顺利完成布局布线并且功能、时序都"符合"要求，最终也还是遇到诸如性能达不到算法要求或者实现的面积过于庞大这样的问题。例子2：在完成方案制定之后，如果直接跳过详细设计的制定，那么编码后的模块间关系很可能显得含糊甚至是混乱。这会增加很多不确定的因素，比如更容易泄漏不易察觉的Bug，或者是直接增加调试阶段定位问题和修改Bug的难度，或者是编码进行到中途发现模块间关系过于混乱复杂，难以保证后续的编码质量，甚至是难以进行后续的工作……因此，越前期的工作步骤对项目的成功有越重要的作用。建立完善的开发流程是很有必要的，而按照开发流程进行工作则很重要。

1.6 一个使用FPGA的经典实例

前面基本上概括了FPGA的特点及其与MCU、DSP的不同之处，这里特别举了一个在一些应用场合下面必须使用FPGA而MCU、DSP无法做到的应用。

在大型通信系统设备中，如3G/4G无线基站中的RRU(Radio Remote Unit)和BBU(Base Band Units)，FPGA担当了重要的角色。以RRU为例，其全称是射频拉远单元，FPGA在该设备里主要实现4个关键功能：

① 发射链路处理：把数字基带信号进行上变频(Digital Up Conversion)、波峰因子降低(Crest Factor Reduction)以及数字预失真(Digital Pre-Distortion)等数字信号处理，然后进行D/A转换，最后通过射频功放发射。

② 接收链路处理：接收高速ADC转换后的数字信号，对其进行数字下变频以及数字滤波，然后传输给基带信号。

③ 接口处理：与高速ADC和高速DAC进行接口，通过LVDS/SerDes接口(JESD204B，一个较新的高速ADC/DAC接口协议)与接收部分的模数转换ADC做接口，接收ADC输出的采样数字数据；与发送部分的数模转换DAC做接口，发送数字数据给DAC。

④ 协议处理与CPU接口：实现与BBU(Base Band Unit)的CPRI (The Common Public Radio Interface，通用公共无线接口)或者OBSAI接口(Open Base Station Architecture Initiative，开放式基站结构)，对数据进行成帧/解帧。另外，FPGA接收CPU的配置和管理。