工程设计与分析系列

Verilog HDL
数字系统设计及仿真

于 斌 编著

视频教学

- ★ Verilog HDL——全球使用最广泛的硬件描述语言
- ★ 易学易用、多接口、应用广泛
- ★ 基础知识—工程实例—实验、课程设计
- ★ 大量光盘内容、视频讲解

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

工程设计与分析系列

Verilog HDL 数字系统设计及仿真

于 斌 编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

Verilog HDL 是一种使用广泛的硬件描述语言，目前在国内无论是集成电路还是嵌入式设计的相关专业都会使用到这种硬件描述语言。

市面上介绍 Verilog HDL 的教材非常广泛，各有不同的偏重。本书着重从设计角度入手，每章都力求让读者掌握一种设计方法，能够利用本章知识进行完整的设计，从模块的角度逐步完成对 Verilog HDL 语法的学习，从而在整体上掌握 Verilog HDL 语法。

为了达到这个目的，每章中都会给出使用本章知识完成的实例，按照门级、数据流级、行为级、任务和函数、测试模块、可综合设计和完整实例的顺序向读者介绍 Verilog HDL 的语法和使用方式。书中出现的所有代码均经过仿真，力求准确，配书光盘中有书中所有实例源文件和实例操作的视频讲解。

本书可作为电子、通信、计算机及集成电路设计相关专业的本科生的教材，同时也适合对 Verilog HDL 感兴趣的爱好者或专业人士阅读。

图书在版编目（CIP）数据

Verilog HDL 数字系统设计及仿真 / 于斌编著. —北京：电子工业出版社，2014.3
（工程设计与分析系列）
ISBN 978-7-121-22284-9

Ⅰ. ①V… Ⅱ. ①于… Ⅲ. ①VHDL 语言—仿真程序—程序设计 Ⅳ. ①TP312

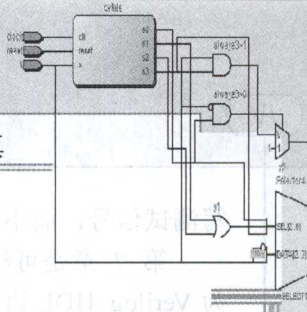
中国版本图书馆 CIP 数据核字（2014）第 001114 号

策划编辑：许存权
责任编辑：刘 凡
印 刷：北京天宇星印刷厂印刷
装 订：三河市皇庄路通装订厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：28.5 字数：725 千字
印 次：2014 年 3 月第 1 次印刷
定 价：65.00 元（含 DVD 光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。



前 言

Verilog HDL 是一门使用非常广泛的硬件描述语言，可以使用在电路和系统级的设计上，也可以作为嵌入式开发的编程语言之一。随着集成电路产业在我国的蓬勃发展，HDL 语言的教学工作也在很多高校展开，市面上也有很多国内外的优秀教材。

作者从事 Verilog HDL 课程的教学多年，使用过十余本中文或译文的教材，然而在教学课程结束之后，学生反馈回来的信息往往是：难以应用。造成这种情况的原因有很多，一是部分教材过于偏重语法细节，在一个细小的语法上纠结许久，使学生陷入了语法大于一切的迷途；二是在学习中与实际电路脱节，写出的代码只适合仿真，殊不知硬件描述语言最终面向的对象还是硬件，只能仿真的代码用途有限；三是缺少直观的认识，对编写的代码、模块等只有纸面上的了解，不去追究其内部的细节。这样学习过 Verilog HDL 语言之后，效果和没学之前相比只是多认识了一些语法而已。

本书在简单地连接了数字电路和 Verilog HDL 的相互关系之后，比较简洁地介绍了基本语法，在介绍语法时给出了范例以使语义明了，并且为每章出现的语法匹配了综合的实例，进一步加深读者的认识。而在语法介绍之后，重点内容放在如何编写可综合的设计模块上，使读者最后编写的模块可以在硬件电路上实现。本书按如下结构进行讲解。

第 1 章是 Verilog HDL 的入门简介，主要回顾了数字电路的设计过程，并介绍使用 Verilog HDL 进行电路设计的基本流程和简单示例，使读者有一个初步的了解。

第 2 章介绍 Verilog HDL 门级建模的基本语法，包括基本的逻辑门、MOS 门和用户自定义原语 UDP 的设计和使用方法，尝试设计一个可以执行的模块并补充了必需的语法，在章节的最后给出了四个门级建模的例子，供读者参考。

第 3 章是数据流建模的相关语法，主要是一些操作数的定义和操作符的使用方法，这些操作数和操作符是 Verilog HDL 的建模基础，在实际的设计中使用频繁，所以在这些语法中给出了很多小例子，在学习时要注意例子间的细小差别。

第 4 章行为级建模也是进行 Verilog HDL 设计的基本语法，主要介绍了 initial 和 always 结构在电路中的使用情况，以及一些语句如 if 语句、case 语句、for 语句、while 语句和 repeat 语句的使用方法，讲解了顺序块和并行块的适用情况，并介绍了命名块和块的禁用语法，最后通过几个实例用这些语法进行电路设计。

第 5 章主要是函数和任务的介绍。函数和任务是 Verilog HDL 中的重要组成部分，它们是一些具有实际功能的代码片段，类似于子程序，可以在 Verilog HDL 的代码中直接调用，非常灵活、方便。另外，编译指令是仿真中的重要指令，也需要理解其用法。

第 6 章的测试模块编写是从仿真测试的角度，力图用多种方式生成不同的信号，给出了同一种信号的多种表达形式，开阔读者的设计思路，使读者能够按照自己习惯的思路来编

写测试信号，而不是局限在某一两种写法上。

第 7 章是可综合模型的设计。从这章开始所有的模块都是可以综合成最终电路的，因为 Verilog HDL 语言就是要编写可以生成实际电路的模块代码。可综合模型设计中需要注意许多问题，如阻塞和非阻塞赋值、多驱动问题、敏感列表问题等，还有一些语法根本不可以综合，本章中也一一列出。最后介绍了流水线的基本思想，并给出了一个雏形。

第 8 章有限状态机的设计是时序电路设计的核心，越是大型的时序电路就越显得状态机非常重要。本章不仅介绍了 moore 型和 mealy 型状态机的区别，给出了一段式、两段式和三段式的写法，更从硬件电路的角度对状态机不同写法得到的电路信号变化做出解释，使读者进一步明白所写模块变成电路后的工作状况。

第 9 章给出了一些常见功能电路的 HDL 模型，一方面让读者对这些功能电路有一定的了解，另一方面也是希望读者能在这些例子中进一步学习 Verilog HDL 编写模块的设计方法。

第 10 章是三个综合的实例，从设计的提出开始到最后的时序仿真结束，完成前端设计的基本流程，让读者有一个整体的流程认知。

第 11 章的实验部分采用了比较新颖的方式，每个实验都有一个主题，在完成这个主题的过程中需要读者编写一些代码，同时也给出了参考代码。读者一方面可以通过这些实验来完成一些实例的设计，另一方面在设计中也可以进一步掌握实验中涉及的语法。

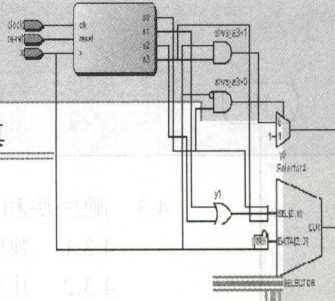
第 12 章的课程设计是一些规模中等的设计模块，每个题目都给出了设计要求、实现代码和仿真结果，部分题目还给出了引脚配置，每个题目的最后都提出了一些问题，还给出了功能扩展建议，当学生觉得题目简单想要加大难度时可以使用这些扩展功能。

附录 A 中给出了测试题，可以检查读者的掌握情况。附录 B 中给出了习题及测试题的答案。

在学习 Verilog HDL 的过程中，一定要注意多编写代码，多进行仿真，这可以帮助读者更好地掌握语法和设计思想。另外，如果有条件的话，建议使用一些 FPGA 或 CPLD 的开发板，把设计的模块用开发板实现，这对读者的学习是非常有益的。

本书由于斌编写第 1～10 章和测试题部分，第 11 章实验部分和第 12 章课程设计部分由杨光编写整理。参与本书编写和光盘开发的人员还有谢龙汉、林伟、魏艳光、林木议、王悦阳、林伟洁、林树财、郑晓、吴苗、莫衍、耿煜、尚涛、李翔、朱小远、唐培培、邓奕、张桂东、鲁力、刘文超、刘新东、米秀杰。书中的代码都经过了编译和仿真，力求准确，但错漏之处难以避免，敬请广大读者批评指正。读者可通过电子邮件 yubin@hrbust.edu.cn 或者 tenlongbook@163.com 与我们交流。

编著者



目 录

第 1 章 Verilog HDL 入门简介 …… 1

1.1 从数字电路讲开来 …… 1
1.2 设计一个七进制计数器 …… 2
1.3 Verilog HDL 建模 …… 4
1.4 集成电路设计流程简介 …… 6
1.5 编写测试代码并仿真 …… 7
1.6 两种硬件描述语言 …… 9

第 2 章 Verilog HDL 门级建模 …… 10

2.1 门级建模范例 …… 10
2.2 门级建模基本语法 …… 12
2.2.1 模块定义 …… 12
2.2.2 端口声明 …… 13
2.2.3 门级调用 …… 15
2.2.4 模块实例化 …… 19
2.2.5 内部连线声明 …… 21
2.3 MOS 开关 …… 22
2.4 用户自定义原语 UDP …… 25
2.4.1 UDP 基本规则 …… 25
2.4.2 组合电路 UDP …… 26
2.4.3 时序电路 UDP …… 29
2.5 层次化设计 …… 31
2.6 应用实例 …… 32
实例 2-1——4 位全加器的门级建模 32
实例 2-2——2-4 译码器的门级建模 35
实例 2-3——主从 D 触发器的门级建模 …… 36
实例 2-4——1 位比较器的门级建模 38
2.7 习题 …… 39

第 3 章 Verilog HDL 数据流级建模 …… 41

3.1 数据流级建模范例 …… 41
3.2 数据流级建模基本语法 …… 42
3.3 操作数 …… 43
3.3.1 数字 …… 43
3.3.2 参数 …… 45
3.3.3 线网 …… 47
3.3.4 寄存器 …… 48
3.3.5 时间 …… 49
3.4 操作符 …… 49
3.4.1 算术操作符 …… 49
3.4.2 按位操作符 …… 49
3.4.3 逻辑操作符 …… 50
3.4.4 关系操作符 …… 51
3.4.5 等式操作符 …… 52
3.4.6 移位操作符 …… 52
3.4.7 拼接操作符 …… 53
3.4.8 缩减操作符 …… 53
3.4.9 条件操作符 …… 53
3.4.10 操作符优先级 …… 54
3.5 应用实例 …… 56
实例 3-1——四位全加器的数据流建模 …… 56
实例 3-2——主从 D 触发器的数据流建模 …… 58
实例 3-3——4 位比较器的数据流建模 …… 59
3.6 习题 …… 60

第 4 章 Verilog HDL 行为级建模 …… 61

4.1 行为级建模范例 …… 61
4.2 initial 结构和 always 结构 …… 64
4.2.1 initial 结构 …… 64
4.2.2 always 结构 …… 66

视频教学

4.3 顺序块和并行块……69
4.3.1 顺序块……69
4.3.2 并行块……70
4.3.3 块的嵌套……72
4.3.4 块的命名与禁用……72
4.4 if 语句……73
4.5 case 语句……76
4.6 循环语句……78
4.6.1 while 循环……78
4.6.2 for 循环……79
4.6.3 repeat 循环……80
4.6.4 forever 循环……80
4.7 过程性赋值语句……81
4.7.1 阻塞性赋值语句……81
4.7.2 非阻塞性赋值语句……81
4.8 应用实例……83
实例 4-1——4 位全加器的行为级建模……83
实例 4-2——简易 ALU 电路的行为级建模……84
实例 4-3——下降沿触发 D 触发器的行为级建模……86
4.9 习题……87

第 5 章 任务、函数与编译指令……88

5.1 任务……88
5.1.1 任务的声明和调用……89
5.1.2 自动任务……91
5.2 函数……93
5.2.1 函数的声明和调用……94
5.2.2 自动函数……96
5.2.3 常量函数……97
5.2.4 任务与函数的比较……98
5.3 系统任务和系统函数……98
5.3.1 显示任务$display 和$write……98
5.3.2 探测任务$strobe……101
5.3.3 监视任务$monitor……101
5.3.4 仿真控制任务$stop 和$finish……103
5.3.5 仿真时间函数$time……103
5.3.6 随机函数$random……104
5.3.7 文件控制任务……105
5.3.8 时间检验任务……109
5.3.9 值变转储任务……109
5.4 编译指令……112
5.4.1 `define……112
5.4.2 `include……113
5.4.3 `timescale……115
5.4.4 `ifdef、`else 和`endif……117
5.5 完整的 module 参考模型……118
5.6 应用实例……119
实例 5-1——信号同步任务……119
实例 5-2——阶乘任务……120
实例 5-3——可控移位函数……121
实例 5-4——偶校验任务……123
实例 5-5——算术逻辑函数……124
5.7 习题……125

第 6 章 Verilog HDL 测试模块……127

6.1 测试模块范例……127
6.2 时钟信号……129
6.3 复位信号……131
6.4 测试向量……133
6.5 响应监控……135
6.6 仿真中对信号的控制……138
6.7 代码覆盖……140
6.8 应用实例……141
实例 6-1——组合逻辑的测试模块……141
实例 6-2——时序逻辑的测试模块……143
实例 6-3——除法器的测试模块……146
6.9 习题……149

第 7 章 可综合模型设计……150

7.1 逻辑综合过程……150
7.2 延迟……153
7.3 再谈阻塞赋值与非阻塞赋值……162
7.4 可综合语法……169
7.5 代码风格……170
7.5.1 多重驱动问题……170
7.5.2 敏感列表不完整……171
7.5.3 if 与 else 不成对出现……171
7.5.4 case 语句缺少 default……172

7.5.5　组合和时序混合设计……172
7.5.6　逻辑简化……173
7.5.7　流水线思想……174
7.6　应用实例……177
实例 7-1——SR 锁存器延迟模型·177
实例 7-2——超前进位加法器……179
实例 7-3——移位除法器模型……182
7.7　习题……187

第 8 章　有限状态机的设计……188

8.1　有限状态机简介……188
8.2　两种红绿灯电路的状态机模型…189
8.2.1　moore 型红绿灯……189
8.2.2　mealy 型红绿灯……194
8.3　深入理解状态机……196
8.3.1　一段式状态机……197
8.3.2　两段式状态机……201
8.3.3　三段式状态机……203
8.3.4　状态编码的选择……211
8.4　应用实例……212
实例 8-1——独热码状态机……212
实例 8-2——格雷码状态机……216
8.5　习题……220

第 9 章　常见功能电路的 HDL 模型……221

9.1　锁存器与触发器……221
9.2　编码器与译码器……229
9.3　寄存器……232
9.4　计数器……237
9.5　分频器……241
9.6　乘法器……247
9.7　存储单元……255
9.8　习题……259

第 10 章　完整的设计实例……260

10.1　异步 FIFO……260
10.1.1　异步 FIFO 的介绍与整体结构……260
10.1.2　亚稳态的处理……262
10.1.3　空满状态的判断……263
10.1.4　子模块设计……266
10.1.5　整体仿真结果……274
10.2　三角函数计算器……277
10.2.1　设计要求的提出……277
10.2.2　数据格式……277
10.2.3　算法的选择与原理结构…278
10.2.4　确定总体模块……281
10.2.5　内部结构的划分……281
10.2.6　分频器模块……283
10.2.7　控制模块……283
10.2.8　迭代设计模块……288
10.2.9　功能仿真与时序仿真……302
10.3　简易 CPU 模型……305
10.3.1　教学模型的要求……305
10.3.2　指令格式的确定……306
10.3.3　整体结构划分……307
10.3.4　控制模块设计……308
10.3.5　其余子模块设计……313
10.3.6　功能仿真与时序仿真……317

第 11 章　实验……321

实验一　简单组合逻辑电路设计……321
实验二　行为级模型设计……328
实验三　任务与函数的设计……335
实验四　流水线的使用……339
实验五　信号发生器设计……344
实验六　有限状态机的设计……350

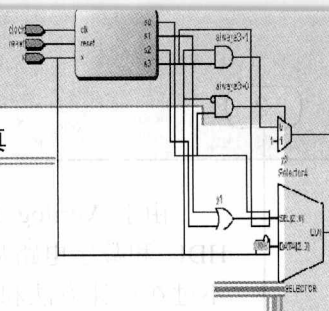
第 12 章　课程设计……356

选题一　出租车计费器……356
选题二　智力抢答器……362
选题三　点阵显示……369
选题四　自动售货机……373
选题五　篮球 24 秒计时……379
选题六　乒乓球游戏电路……384
选题七　CRC 检测……398
选题八　堆栈设计……404
选题九　数字闹钟……410

附录 A　课程测试样卷……419

附录 B　习题及样卷答案……424

视频教学



第 1 章　Verilog HDL 入门简介

数字电路的基本知识是 Verilog HDL 的入门基础，本章中通过回忆数字电路的基本设计流程，引出 Verilog HDL 的设计方法和简单示例，旨在为读者解决使用 Verilog HDL 进行设计的一些入门知识，使读者对 Verilog HDL 有一个初步的认识，并了解 Verilog HDL 与数字电路的关系。请带着如下问题来阅读本章：

（1）Verilog HDL 与数字电路有什么联系？

（2）使用 Verilog HDL 编写出的代码能用在哪里？

（3）采用 Verilog HDL 进行电路设计与传统数字电路设计在流程上是如何对应的？

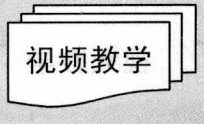
本章内容

- 数字电路设计的回顾
- 使用 Verilog HDL 进行电路设计
- 集成电路设计的基本流程

1.1　从数字电路讲开来

在过去的几十年中，数字电路设计技术发展迅速。从简单的逻辑电路发展到集成电路，直至出现现在主流的超大规模集成电路。设计技术的发展必然带动设计手段的更新，传统的数字电路设计流程也在逐渐发生着改变。一方面，由于设计电路规模的不断扩大，设计人员的人力操作显得越来越单薄，急需计算机的大力辅助，于是促进了电子设计自动化（Electronic Design Automation，EDA）的出现和发展；另一方面，传统的数字电路的基本设计流程也无法应对急速增长的电路规模，面对着上万规模的门级电路，传统的在设计图纸上或计算机上手动完成最终电路图的方法变得越来越难以完成，同时带来的还有测试时的更大难题。于是，迫切需要某种方法，使设计者可以使用 EDA 工具完成这种大规模的集成电路设计。

Verilog HDL 在这种情况下应运而生，Verilog HDL 可以采用编写代码的方式来设计数字电路，向下可以到达底层的门级电路，向上可以抽象到高层的电路行为描述，使得原本需要成百上千的门级电路设计变为几条简单易懂的编程代码，无论是从视觉上、功能上还是后期的检测上，都使数字电路的设计速度有了很大的提升，迅速成为超大规模数字电路设计的标准设计语言。



由于 Verilog HDL 是为了解决传统数字电路设计过程中的瓶颈而产生的，所以 Verilog HDL 和数字电路从最终目的上来讲，都是为了生成一个可以实现某种功能的数字电路，只不过在设计方法和手段上有所区别而已。要学习 Verilog HDL 语言，首先就要弄清楚采用 Verilog HDL 语言进行设计和采用传统设计过程到底是如何对应的，这样可以对 Verilog HDL 有一个最基本的定位。

1.2 设计一个七进制计数器

为了说明使用 Verilog HDL 进行电路设计的设计过程，首先要回忆一下在数字电路课程中学习过的基本设计方法。下面通过一个简单的例子来回顾一下，同时也是为了使读者的头脑中有一个基本的流程。

本例是要设计一个七进制的计数器，在数字电路的相关教材中都有类似的介绍。首先要确定工作状态，画出状态转换图并写出对应的输出。画出的状态转换图如图 1-1 所示，图中从 000 到 110 的循环状态是要设计的七进制循环，即从十进制的 0 计数到 6，一共计 7 个数。由于 3 位二进制数总共可表示 8 个数值，有一个数值 111 并未使用，为了使电路能够自启动，在设计的时候直接把 111 这个状态的下一状态指向 000，这样就出现了图中的状态转换。在箭头上标示出的是每一个状态的输出值，仅在 110 状态的时候电路产生一个进位信号，其他状态下都输出低电平信号。状态编码的时候采用左侧为高位、右侧为低位的方式，即采用图中的 $Q_3Q_2Q_1$ 的顺序进行编码，这个顺序在后面的设计过程中也是需要的。完成上述过程之后，就可以得到图 1-1 中的状态转换图。这也是时序电路设计的基本步骤，状态编码可以采用其他的方式编码，但是所画出的状态转换图与本图大同小异。

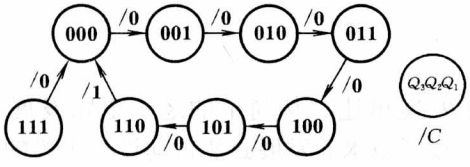


图 1-1 状态转换图

Q_3 \ Q_2Q_1	00	01	11	10
0	001	010	100	011
1	101	110	000	000

图 1-2 整体卡诺图

接下来要画出整体设计的卡诺图，并分别求出每一个输出信号的卡诺图。由图 1-1 可以得到图 1-2 所示的整体卡诺图，由于进位输出信号比较简单，这里没有标示在卡诺图中。卡诺图的外侧是当前的状态值，内部为当前状态所指向的下一状态。得到此图后可以进一步拆分，得到图 1-3 所示的每一个状态位的卡诺图。

Q_3 \ Q_2Q_1	00	01	11	10
0	0	0	1	0
1	1	1	0	0

（a）Q_3* 的卡诺图

Q_3 \ Q_2Q_1	00	01	11	10
0	0	1	0	1
1	0	1	0	0

（b）Q_2* 的卡诺图

Q_3 \ Q_2Q_1	00	01	11	10
0	1	0	0	1
1	1	0	0	0

（c）Q_1* 的卡诺图

图 1-3 三个状态位的卡诺图

第三步要在卡诺图中圈 1 得到输出的状态方程，可以得到式 1-1 的状态方程::

$$\begin{cases} Q_3^* = Q_3Q_2' + Q_3'Q_2Q_1 \\ Q_2^* = Q_2'Q_1 + Q_3'Q_2Q_1' \\ Q_1^* = Q_2'Q_1' + Q_3'Q_1 \end{cases} \tag{1-1}$$

使用不同的触发器需要对该状态方程做不同的转换，这里使用 JK 触发器来设计此计数器，由于 JK 触发器的特性方程为 $Q^* = JQ' + K'Q$，所以要把式（1-1）中的状态方程转化成和特性方程相同的形式，可以得到式（1-2）：

$$\begin{cases} Q_3^* = Q_2Q_1 \cdot Q_3' + Q_2' \cdot Q_3 \\ Q_2^* = Q_1 \cdot Q_2' + Q_3'Q_1' \cdot Q_2 = Q_1 \cdot Q_2' + (Q_3 + Q_1)' \cdot Q_2 \\ Q_1^* = (Q_2' + Q_3') \cdot Q_1' = (Q_3Q_2)' \cdot Q_1' + 1' \cdot Q_1 \end{cases} \tag{1-2}$$

由式（1-2）可以得知 JK 触发器的驱动方程，得到式（1-3）：

$$\begin{cases} J_3 = Q_2Q_1, K_3 = Q_2 \\ J_2 = Q_1, K_2 = Q_3 + Q_1 \\ J_1 = (Q_3Q_2)', K_1 = 1 \end{cases} \tag{1-3}$$

这样，每个触发器的驱动信号就得到了。同时，可以知道该计数器的进位输出信号在 110 状态时输出高电平，可得到输出方程为式（1-4）：

$$C = Q_3Q_2Q_1' \tag{1-4}$$

得到这些式子之后，就可以画出图 1-4 所示的电路图。按照图中的电路进行连接后，就可以进行功能测试，得到最终的验证结果。再按照此电路图完成元器件的连接，就能够产生一个七进制的计数器了。到此为止，一个简单的数字电路设计就结束了。

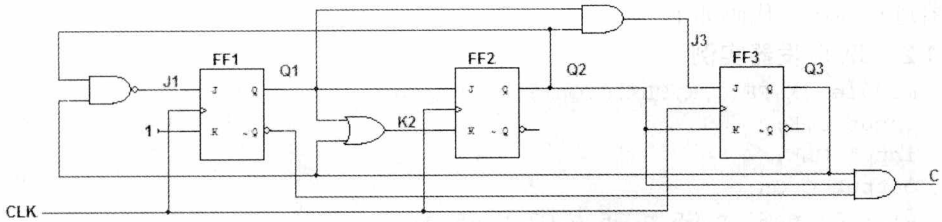


图 1-4　七进制计数器电路图

由上述例子可以看到，从设计的基本功能要求到最终完成一个可以实现功能的电路图，中间大约有四个大步骤：状态图、化简、驱动的确定和最终电路的连接，这几个部分缺一不可。本例是一个仅有 4 位输出一个时钟输出的简单例子，实际的功能电路规模和复杂程度要大大超过本例，即使采用某些功能模块的直接使用，电路的规模也是一个非常庞大的数量。例如比较古老的 486 游戏机，其内部逻辑门也达到几十万个。且不说状态图是否易画，单单一个化简工作就非常难以实现，更不要说后面的驱动和电路的连接更是千头万绪，一旦出错，既不易查找也不易修改。

那么，采用 Verilog HDL 来进行设计，就可以绕开这些问题了么？

1.3 Verilog HDL 建模

我们暂且不去讨论 Verilog HDL 的优越性，仅把 1.2 节中的例子采用 Verilog HDL 来实现。先来看一个比较容易理解、但是却比较麻烦的例子。代码如下：

例 1.1 七进制计数器实例

```
module Counter(Q3,Q2,Q1,C,CLK);
output Q3,Q2,Q1,C;
input CLK;
wire J1,K2,J3;

JK FF  JK1(Q1,Q1n,J1,1,CLK);
JK FF  JK2(Q2, ,Q1,K2,CLK);
JK FF  JK3(Q3, J3,Q2,CLK);

and and1(C,Q3,Q2,Q1n);
and and2(J3,Q1,Q2);
nand nand1(J1,Q2,Q3);
or or1(K2,Q1,Q3);

endmodule
```

上述代码和图 1-4 中的各个电路符号一一对应，称为设计模块，代码中的 and、nand、or 就是图中的与门、与非门和或门，JK_FF 就是图 1-4 中的 JK 触发器（本章中的代码无须看懂，只要知道是描述该电路即可，具体的语法会在后续章节中介绍）。

JK 触发器在实际电路中是可以直接使用已有电路的，但在此代码中需要再对 JK_FF 这个触发器进行描述，代码如下。

例 1.2 JK 触发器实例

```
module JK FF(J,K,CLK,Q,Qn);
input J,K;
input CLK;
output Q,Qn;
wire G3 n,G4 n,G5 n,G6 n,G7 n,G8 n;

nand G7(G7 n,Qn,J,CLK);
nand G8(G8 n,CLK,K,Q);
nand G5(G5 n,G8 n,G6 n);
nand G6(G6 n,G5 n,G8 n);
nand G3(G3 n,G5 n,CLK n);
nand G4(G4 n,CLK n,G6 n);
nand G1(Q,G3 n,Qn);
nand G2(Qn,Q,G4 n);

not G9(CLK n,CLK);

endmodule
```

该代码所描述的电路如图 1-5 所示，这是一个最基本的上升沿触发的 JK 触发器的电路

图，主要使用到了与非门和非门两种基本逻辑电路，具体的对应关系代码中已经给出了。完成了上述两个代码，一个七进制计数器的 Verilog HDL 设计就结束了。

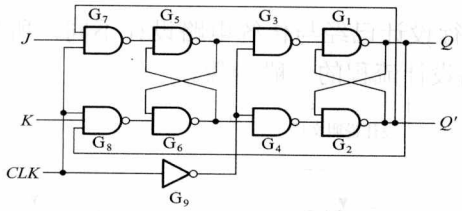


图 1-5　JK 触发器电路图

如果单从上面的代码来看，使用 Verilog 描述并不比前面的电路设计方法更加简洁。可以看看下面的代码，此代码描述的依然是一个七进制计数器。

例 1.3　七进制计数器另一种建模方法

```
module Counter(Q,CLK,RESET);
output [2:0] Q;
input CLK,RESET;
reg [2:0] Q;

always @ (posedge CLK)
if(RESET)
  Q<=0;
else if (Q==6)
  Q<=0;
else
  Q<=Q+1;

endmodule
```

此代码就是从行为级的角度描述了一个功能电路。如果把 1.2 节中的电路规模扩大，如扩大到 64 进制的计数器，那么相对应的化简、连接电路等工作量就会呈几何级数增长。但是对于上面的 Verilog HDL 代码来说，仅仅需要修改几个字母和数字而已。当然，设计的复杂性不能这么简单地计算，这里只是想通过一个例子来告诉大家：Verilog HDL 在应对大规模集成电路设计方面确实有它自己的优势。

Verilog HDL 作为一种设计语言，编写出的代码需要进行一定的转化才能变成实际可实现的电路。而且这里要特别强调一点，使用 Verilog HDL 描述的代码最终要转化成和图 1-4 类似的电路形式才能最终实现。因为有些使用 FPGA 或 CPLD 进行设计开发的爱好者会忽略这个问题。例如，上面的七进制计数器代码经转化后会得到图 1-6 所示的电路，这个电路无论是采用集成电路设计还是直接使用通用模块搭建，都是可以实现其最终功能的。

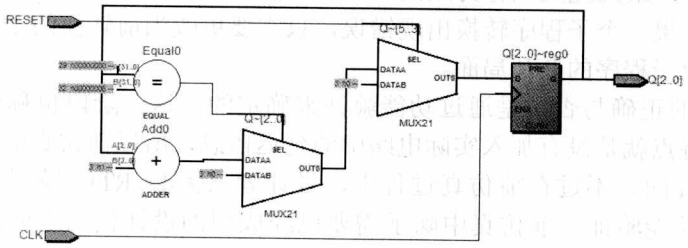


图 1-6　转化为可实现的电路图

1.4 集成电路设计流程简介

采用 Verilog HDL 进行设计已经与传统电路设计不同，所以它的设计流程也会发生改变，图 1-7 所示为集成电路设计流程的一般步骤。

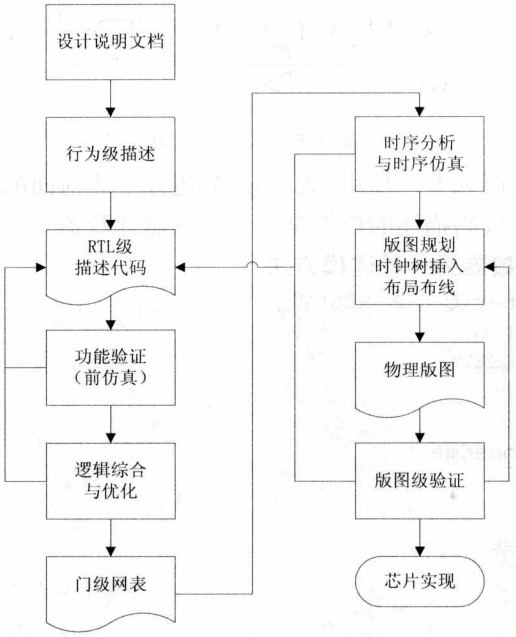


图 1-7 集成电路设计一般流程

设计的最开始阶段一定是设计文档的编写。这个设计说明文档主要包含了设计要实现的具体功能和期待实现的详细性能指标，包括电路整体结构、输入/输出 I/O 接口、最低工作频率、可扩展性等参数要求。完成设计说明文档后，需要用行为级描述待设计的电路。行为级描述可以采用高级语言，如 C/C++等，也可以采用 HDL 来编写。这个阶段的描述代码并不要求可综合，只需要搭建出一个满足设计说明的行为模型即可。

行为级描述之后是 RTL 级描述。这一阶段就采用硬件描述语言来编写，一般采用 VHDL 或 Verilog HDL 来实现，对于两者的联系也会在后续章节中简单介绍。对于比较大的设计，一般是在行为级描述时采用 C/C++搭建模型，在 RTL 级描述阶段，逐一对行为模型中的子程序进行代码转换，用 HDL 代码取代原有的 C/C++代码，再利用仿真工具的接口，将转换成 HDL 代码的子程序加载到行为模型中，验证转换是否成功，并依次转换行为模型中的所有子程序，最终完成从行为级到 RTL 级的 HDL 代码描述。这样做的好处是减少了调试的工作量，如果一个子程序转换出现错误，只需要更改当前转换的子程序即可，避免同时出现多个待修改子程序的杂乱局面。

RTL 模型的正确与否，是通过功能验证来确定的，这一阶段也称前仿真或功能仿真。前仿真的最大特点就是没有加入实际电路中的延迟信息，所以前仿真的结果与实际电路结果还是有很大差异的。不过在前仿真过程中，设计者只关心 RTL 模型是否能完成预期的功能，所以称为功能验证。前仿真中除了需要已经成型的设计代码之外，还需要一个验证环境，这个验证环境也可以使用 Verilog HDL 语言来搭建，在本书中也会有介绍。

当 RTL 模型通过功能验证后，就进入逻辑综合与优化阶段。这个阶段主要是由 EDA 工具来完成的，设计者可以给综合工具指定一些性能参数、选择一些工艺库等，使综合出来的电路符合自己的要求。这个过程也就是 1.3 节中把计数器的 Verilog HDL 代码变为图 1-6 所示电路图的过程。

综合生成的文件是门级网表，这个网表文件包含了综合之后的电路信息，还会根据工艺库的不同得到设计的延迟信息。将这些延迟信息反标注到 RTL 模型当中，进行时序分析，主要检测的是建立时间（Setup Time）和保持时间（Hold Time）。其中，建立时间的违例和保持时间较大的违例必须要修正，可以采用修正 RTL 模型或修改综合参数来完成。对于较小的保持时间违例，可以放到后续步骤中修正。对综合之后包含延迟信息的门级网表模型进行仿真验证的过程称为时序仿真，时序仿真的结果更加逼近实际电路。到此步骤为止，硬件描述语言的部分就结束了，剩下的步骤就要进入电路的版图布局阶段了。

设计通过时序分析后，就可以进行版图规划与布局布线了。这个阶段是把综合后的电路按一定的规则进行排布，设计者也可以添加一些参数对版图的大小和速度等性能进行约束。布局布线的结果是生成一个物理版图，再对这个版图进行仿真验证，如果不符合要求，就需要向上查找出错点，重新布局布线或修改 RTL 模型。如果版图验证符合要求，这个设计就可以送到工艺生产线上，进行实际芯片的生产。

当然，上述流程只是一个基本的过程，其中很多步骤都是可以展开成很多细小的步骤，也有一些步骤（如形式验证）在这个流程中并没有体现。不过这个流程图可以包含基本的 IC 步骤，对于初学者已经足够了。另外，不同的公司推荐流程不同的原因是采用了不同的 EDA 软件来完成上述 IC 基本流程，一般来说比较大的 EDA 软件公司都会有自己完整的一套 IC 设计流程，但步骤大同小异。例如前仿真阶段，可用于 HDL 仿真的 EDA 工具就有 Synopsis 公司的 VCS、Cadence 公司的 Verilog-XL、明导公司的 ModelSim 等。

1.5 编写测试代码并仿真

按照前面介绍的基本设计流程，在设计代码编写结束后就需要进行功能验证，即前仿真。所要进行的操作也是使用 Verilog HDL 描述一个代码，这个代码的功能是给前面写好的设计代码提供一系列变化的输入激励。就如同给灯泡加上电压才知道灯泡会不会亮一样，设计好的代码也需要加上所需要的条件才能知道是否能正常工作，这个条件一般称为输入激励或测试向量，目的就是为了模拟该设计实际工作时的输入条件，来测试一下设计在不同状态下的工作状况，其反应在代码中就是观察设计的输出值是否正确。例如，对七进制计数器的就可以使用如下的测试代码。

例 1.4　七进制计数器测试代码

```
module Test Counter;
reg CLK,RESET;
wire [2:0] Q;
//以下为测试激励
initial
begin
RESET<=1;
# 50 RESET<=0;
# 1000 RESET<=0;
```

```
    end

    initial
    CLK<=0;
    always #5 CLK<=!CLK;
    //以下是模块调用
    Counter Counter(Q,CLK,RESET);

    endmodule
```

由于设计的输入仅有两个 1 位信号，所以在代码中的测试激励部分仅对两个输入信号的变化情况进行描述。例如“RESET<=1”等代码，是给该信号加上高电平，其中的 1 和 0 与数字电路中无异，就是代表高低电压。给设计代码加上了高低电压之后看设计的 Counter 电路是否正常输出，就要看 Q 的输出是否正常，因为 Q 是输出，所以不需要加上 1、0 信号，仅需要采用连线的方式来查看即可。当然，所有的这些输入/输出都需要有一个设计模块来连接，就像有了电池需要拿灯泡来检验一样，代码中的模块调用部分的功能就是“拿”来一个已经写好的设计，“放”在当前的工作环境里，测试它是否正常，具体的语法会在后面介绍。

测试激励和模块调用，这两个部分是最主要的。一般把包含这两个部分的代码称为测试模块，也可以称为测试平台或测试环境，一些书中所说的要搭建一个测试平台，意思就是要给当前设计编写一个比较好的测试模块，当然在编写的时候有很多注意的事项，也有很多常用的写法，这里暂且不予讨论。

编写测试模块后就可以对设计模块进行仿真，前仿真是没有时间延迟的，也就是说电路的信号变化都是在瞬间完成的。例如图 1-8 所示的仿真波形结果，每次当第一行的 CLK 信号变化时，最后一行的输出值 Q 都会立刻随着变化，没有任何的时间间隔。这在实际电路中是根本不可能的，所以此步骤称为功能仿真，仅仅查看功能，而不考虑实际工作的状况。由图 1-8 易知，在 RESET 为高电平的时候，该电路维持在 0 值不变，在 RESET 为低电平的时候，每次 CLK 从低电平变为高电平（即上升沿）Q 值就做加 1 操作，循环过程为 000 至 110，共计 7 个数，功能与之前设想的完全一致，此时功能仿真已经成功。

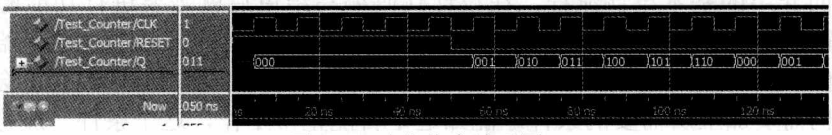


图 1-8　功能仿真波形图

接下来就可以进行综合，综合的过程介绍起来比较烦琐，本章中暂不介绍。在代码综合之后还可以进行时序仿真，时序仿真的结果就比较贴近实际工作的状态了。如图 1-9 所示就是时序仿真的波形结果，可以看到功能还是和图 1-8 相同，输出值也相同，只是每一个输出的 Q 值变化的位置发生了改变，图 1-8 中是在 CLK 的上升沿变化的，而图 1-9 中要滞后于 CLK 的上升沿。如果从第一次 Q 值变为 001 算起的话，图 1-8 是在第 55ns 处，而图 1-9 是在 60ns 之后。其原因就是由于时序仿真加入了电路的实际延迟信息，模拟了实际工作状态，所以时序仿真更加接近实际电路的工作状态。图 1-10 还截取了部分图像，可以看到 Q 值在变化为 110 信号的时候还有一个中间的变化状态，这也是区分时序仿真和功能仿真的一个细节，因为时序仿真波形图中往往会有很多的中间态，这些状态是无用的，只有最后稳定的值才是有效值。

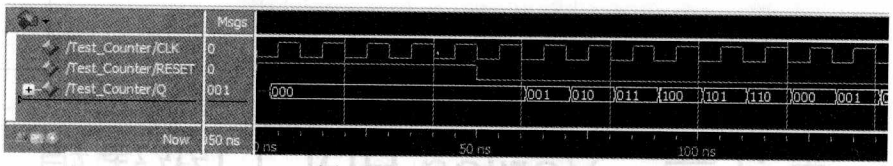


图 1-9　时序仿真波形图

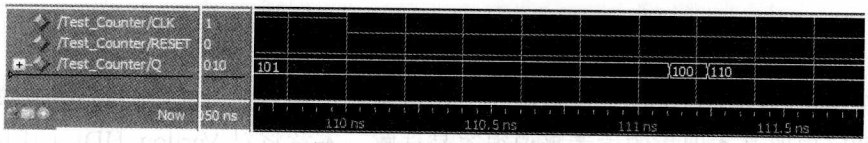


图 1-10　时序仿真细节

完成了上述步骤，使用 Verilog HDL 的部分也就结束了。本书关注的就是从设计代码的编写到测试模块的编写，以及如何更好地实现设计和仿真的功能。

1.6　两种硬件描述语言

既然说到了 Verilog HDL，就不得不说一下 VHDL，因为二者都是硬件描述语言，HDL 就是 Hardware Discription Language（硬件描述语言）的缩写，两者也都是为了设计电路而产生的硬件描述语言。

Verilog HDL 是一种以文本形式来描述数字系统硬件的结构和行为的语言，用它可以表示逻辑电路图、逻辑表达式，还可以表示数字逻辑系统所完成的逻辑功能。它是在用途最广泛的 C 语言的基础上发展起来的一种硬件描述语言，是由 GDA(Gateway Design Automation) 公司的 Phil Moorby 在 1983 年末首创的，最初只设计了一个仿真与验证工具，之后又陆续开发了相关的故障模拟与时序分析工具。1985 年 Moorby 推出它的第三个商用仿真器 Verilog-XL，获得了巨大的成功，从而使得 Verilog HDL 迅速得到推广应用。1989 年 CADENCE 公司收购了 GDA 公司，使得 Verilog HDL 成为了该公司的独家专利。1990 年 CADENCE 公司公开发表了 Verilog HDL，并成立 LVI 组织以促进 Verilog HDL 成为 IEEE 标准，即 IEEE Standard 1364—1995，后又发展为 1364—2001 标准，在其基础上还建立了验证功能更强的 System Verilog（即 SV 语言）。 Verilog HDL 的最大特点就是易学易用，如果有 C 语言的编程经验，可以在一个较短的时间内很快地学习和掌握。但 Verilog HDL 的语法比较自由，也容易造成初学者犯一些错误，这一点要注意。

VHDL 全名是 Very-High-Speed Integrated Circuit HardwarvDescription Language，诞生于 1982 年。1987 年底，VHDL 被 IEEE 和美国国防部确认为标准硬件描述语言。自 IEEE-1076（简称 87 版）之后，各 EDA 公司相继推出自己的 VHDL 设计环境，或宣布自己的设计工具可以和 VHDL 接口。1993 年，IEEE 对 VHDL 进行了修订，从更高的抽象层次和系统描述能力上扩展 VHDL 的内容，公布了新版本的 VHDL，即 IEEE 标准的 1076—1993 版本，简称 93 版。VHDL 和 Verilog 作为 IEEE 的工业标准硬件描述语言，得到了众多 EDA 公司的支持。

简单来说，Verilog HDL 易于上手，现在公司中大多采用 Verilog HDL 也是因为容易入门，但是编写代码时要注意代码风格，也要考虑对综合的影响；VHDL 学习起来相对较慢，但所写的代码和综合后的实际电路基本相符，不容易出现太大的偏差，而且国内高校选择 VHDL 教学的较多。读者可以根据实际情况进行选择。