工程设计与分析系列

Verilog HDL（第2版）
数字系统设计及仿真

于 斌 黄 海 编著

视频教学

本书程序代码资源及视频下载：
www.hxedu.com.cn

- ★ Verilog HDL——使用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
- ★ 海量素材、视频讲解

中国工信出版集团
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（第2版）

于 斌 黄 海 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的相关专业都会用到这种硬件描述语言，市面上有关 Verilog HDL 的教材也比较多，但各有不同的偏重。

本书在第 1 版广泛应用的基础上，吸收了众多读者的宝贵建议，大幅完善了内容。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，能够利用该章知识进行完整设计，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。

为了达到上述目的，每章都给出使用该章知识完成的实例，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 的语法和使用方法。书中出现的所有代码均经过仿真测试，力求准确，另外配套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、计算机和集成电路相关专业的本科生教材，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，黄海编著. —2 版. —北京：电子工业出版社，2018.1
（工程设计与分析系列）
ISBN 978-7-121-33010-0

Ⅰ. ①V… Ⅱ. ①于… ②黄… Ⅲ. ①硬件描述语言—程序设计 Ⅳ. ①TP312

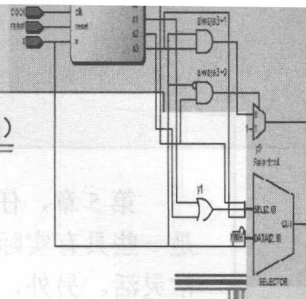
中国版本图书馆 CIP 数据核字（2017）第 275041 号

策划编辑：许存权（QQ：76584717）
责任编辑：许存权　特约编辑：谢忠玉　等
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
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：28.75　字数：740 千字
版　　次：2014 年 3 月第 1 版
　　2018 年 1 月第 2 版
印　　次：2018 年 1 月第 1 次印刷
定　　价：69.00 元

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）88254888，88258888。

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。

本书咨询联系方式：（010）88254484，xucq@phei.com.cn。



再 版 前 言

Verilog HDL 是一种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 课程教学多年，使用过十余种本版和引进版的教材，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太多，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，不知硬件描述语言最终面向的对象是硬件，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 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。

自本书第 1 版 2014 年出版以来，获得读者的广泛欢迎，已多次重印，并且，很多读者来信介绍他们具体应用 Verilog HDL 的情况，对本书提出了很多宝贵意见和建议。在此基础上，我们根据用户建议，结合相关企业应用的需求和高校教学需求修订了第 1 版内容。相对于第 1 版本，本书删减了一些使用频率较低的语法，降低了读者掌握语法的难度，同时增加了一些实例，使读者有更多可以学习和揣摩的范例，能更好地理解代码的设计。

本书在简单地介绍了数字电路和 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实例，使读者进一步加深认识。而在介绍语法之后，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展开。

第 1 章，Verilog HDL 入门简介。主要回顾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，Verilog HDL 门级建模。介绍 Verilog HDL 门级建模的基本语法，主要讲解基本逻辑门的使用方法和层次化建模思想，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实例，供读者参考。

第 3 章，Verilog HDL 数据流级建模。介绍数据流级建模的相关语法，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，Verilog HDL 行为级建模。行为级建模，也是进行 Verilog HDL 设计的基本语法，主要介绍 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 语句和循环语句，讲解顺序块和并行块的适用情况，并介绍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，任务、函数与编译指令。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代码中直接调用，非常灵活。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，Verilog HDL测试模块。从仿真测试的角度编写测试模块，力图用多种方式生成不同信号，给出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，而不是局限在某一种写法上。

第7章，可综合模型设计。从本章开始，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，有限状态机设计。状态机的设计是时序电路设计的核心，越大型的时序电路状态机就越显得重要。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，而且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进行解释，使读者更明白所写模块变成电路后的工作状况。

第9章，常见功能电路的HDL模型。本章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，完整的设计实例。本章有三个综合实例，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，使读者有一个整体的流程认知。

第11章，实验。本章有七个实验，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。每个实验都分成了学生版和辅导版两个部分，学生版可以直接在实验中给学生使用，辅导版则可以给教师作为参考或学生自学辅导使用。

第12章，课程设计。本章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。

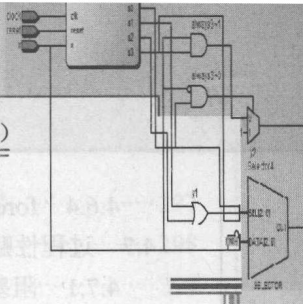
附录A，课程测试样卷。给出了测试题，可以检查读者的掌握情况。

附录B，习题及样卷答案。给出了习题和测试题的答案，以便参考。

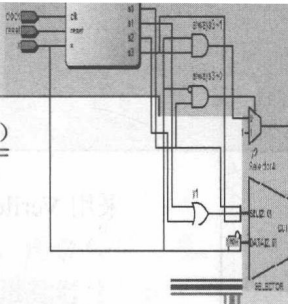
在学习Verilog HDL的过程中，一定要多编写代码，多进行仿真，这样可以帮助读者更好地掌握语法和设计思想。另外，如果有条件的话，建议使用一些FPGA或CPLD的开发板，把设计的模块用开发板实现，对读者的学习非常有益。

本书第1～6章和测试题部分由哈尔滨理工大学于斌编写，第7~12章由哈尔滨理工大学黄海编写。参与本书编写和视频开发的人员还有谢龙汉、蔡思祺、林伟、魏艳光、林木议、王悦阳、林伟洁、林树财、郑晓、吴苗、李翔、朱小远、唐培培、耿煜、尚涛、邓奕、张桂东、鲁力等。由于时间仓促，书中疏漏之处，请读者批评指正，可通过电子邮件yubin@hrbust.edu.cn与我们交流。本书配套素材光盘内容。请在华信教育资源网（www.hxedu.com.cn）的本书页面下载，或与本书作者和编辑联系。

编著者



目　录



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，本章中通过回顾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？

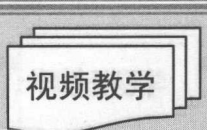
本章内容

- 集成电路设计的基本流程
- 数字电路设计示例
- Verilog HDL 电路设计示例

1.1　集成电路设计流程简介

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提升，迅速成为超大规模数字电路设计的标准设计语言。



采用 Verilog HDL 进行设计的基本流程如图 1-1 所示。

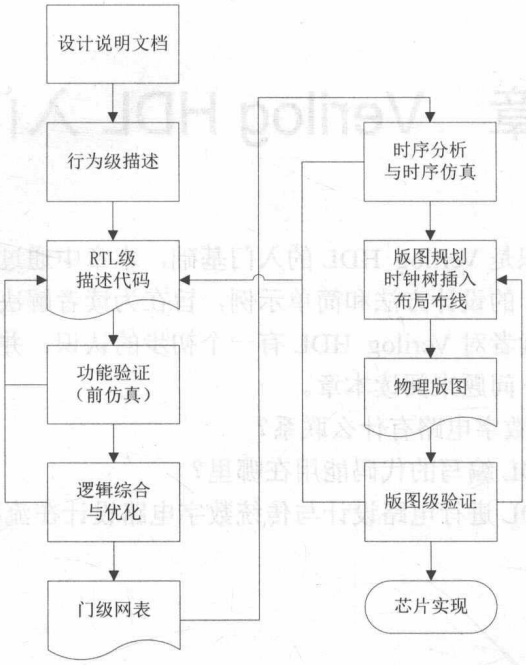


图 1-1　集成电路设计一般流程

设计的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，这一阶段也称为前仿真或功能仿真。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有很大差异。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。

综合生成的文件是门级网表，这个网表文件包含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。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。

1.2 数字电路设计范例

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，就要清晰把握 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，这样在使用 Verilog HDL 时会有一个最基本的定位。

本节通过一个典型的数字电路例子来回忆一下数字电路的基本设计方法，也让读者的头脑中有一个基本的流程。

例题：设计一个七进制计数器。

解答：（1）画出状态转换图。

画出的状态转换图如图 1-2 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。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。完成上述过程之后，就可以得到图 1-2 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。

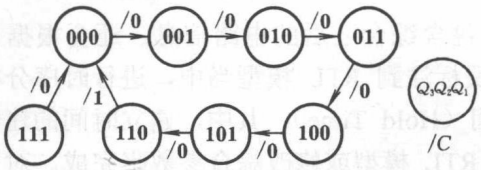


图 1-2　状态转换图

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

图 1-3　整体卡诺图

（2）列出输入输出的卡诺图。

由图 1-2 可以得到图 1-3 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 1-4 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 1-4　三个状态位的卡诺图

（3）写出状态方程。

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。

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

（4）整理得到驱动方程和输出方程。

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）。

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

（5）画出电路图并测试功能。

得到输出方程和驱动方程后，就可以画出如图 1-5 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。

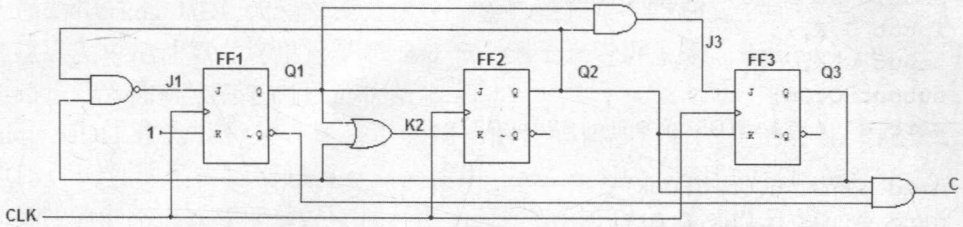


图 1-5　七进制计数器电路图

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 位输出和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，仅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。

1.3　Verilog HDL 建模范例

上一节中的例子如果采用 Verilog HDL 来实现，有多种实现方法。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 1-5 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，只作为示例给出）。

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 1-6 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。完成了上述两个代码，一个七进制计数器的 Verilog HDL 设计就完成了。

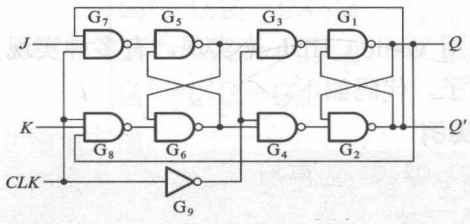


图 1-6　JK 触发器电路图

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。下例依然是一个七进制计数器的 Verilog HDL 模型。

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 描述的代码最终依然要转化成和图 1-5 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 1-7 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。

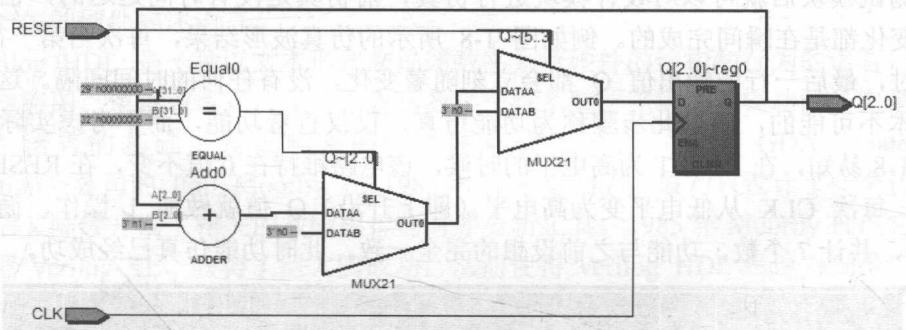


图 1-7　转化为可实现的电路图

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、是不是能工作一样，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，设计的输出值是否正确。

例如，七进制计数器的工作状况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。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，具体的语法会在第 6 章介绍。

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。

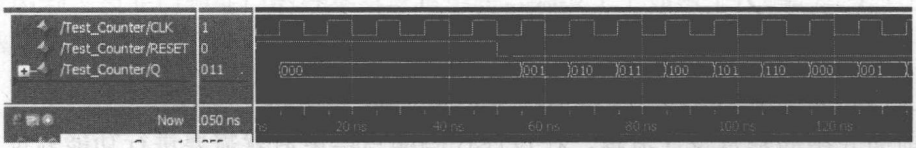


图 1-8　功能仿真波形图

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，时序仿真的结果比较贴近实际工作的状态。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。

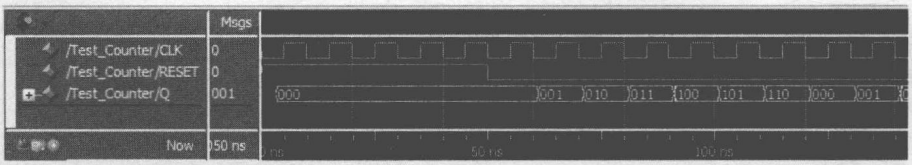


图 1-9　时序仿真波形图

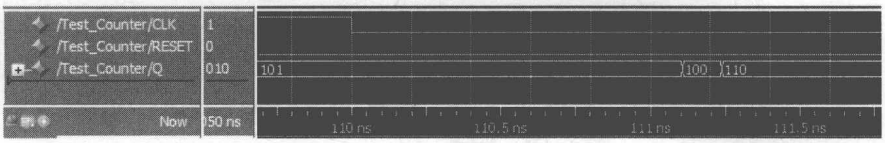


图 1-10　时序仿真细节

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。

1.4　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）的首字母缩写，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 也是因为容易入门之故，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。