基于Quartus Ⅱ的数字逻辑电路设计技术基础

王书志
编著

兰州大学出版社

基于Quartus Ⅱ的数字逻辑电路设计技术基础

王书志
编著

兰州大学出版社

图书在版编目(CIP)数据

基于 Quartus Ⅱ的数字逻辑电路设计技术基础/王书志编著. —兰州:兰州大学出版社,2011.5

ISBN 978-7-311-03686-7

Ⅰ.①基… Ⅱ.①王… Ⅲ.①数字电路:逻辑电路②硬件描述语言,Verilog HDL—程序设计 Ⅳ.①TN79②TP312

中国版本图书馆 CIP 数据核字(2011)第 084172 号

策划编辑 张爱民
责任编辑 郝可伟
封面设计 刘 杰

书 名 基于 Quartus Ⅱ的数字逻辑电路设计技术基础
作 者 王书志 编著
出版发行 兰州大学出版社 (地址:兰州市天水南路 222 号 730000)
电 话 0931-8912613(总编办公室) 0931-8617156(营销中心)
0931-8914298(读者服务部)
网 址 http://www.onbook.com.cn
电子信箱 press@lzu.edu.cn
印 刷 兰州德辉印务有限责任公司
开 本 787×1092 1/16
印 张 10.5
字 数 251 千
版 次 2011 年 5 月第 1 版
印 次 2011 年 5 月第 1 次印刷
书 号 ISBN 978-7-311-03686-7
定 价 18.00 元

(图书若有破损、缺页、掉页可随时与本社联系)

内容简介

本书共分六章，第一章介绍了 Verilog HDL 的基本知识、模块、运算符、数据类型及基本词法等，第二章介绍了利用 Quartus Ⅱ 9.0 进行电路设计的方法，第三章给出了一些常见的组合逻辑电路设计方法以及设计练习，第四章给出了触发器的设计方法，第五章给出了时序逻辑电路的设计方法及设计练习，第六章为综合电路的设计。

本书通俗易懂，可以作为普通高等学校数字逻辑及系统设计教材、EDA 课程的教材，也可供数字系统设计的工程技术人员参考。

前 言

随着电子技术的迅猛发展,传统的用74系列器件和面包板来设计数字电路和系统的方法正逐步退出历史舞台,取而代之的是采用可编程逻辑器件、EDA设计软件和硬件描述语言的设计技术。大规模可编程现场门阵列(FPGA)和复杂的可编程器件(CPLD)是当今广泛应用的两类主流器件,用它们设计的产品,体积小,功耗小,速度快,可靠性高,容量大,使用灵活,开发周期短,成本低,系统的硬件功能可以像软件一样通过编程的方式随意改变,从而极大地提高了设计的灵活性、方便性。目前,可编程逻辑器件(PLD)广泛应用于航空航天、军事、工业控制、仪器仪表、家用电器、手机、网络通信、计算机等各个领域。

FPGA和CPLD得到空前发展以及广泛应用,很多用人单位也对该方面的知识有需求。因此,很多人渴望掌握这方面的知识和设计方法。从另一个方面看,上述现实也对目前的数字电子技术教学体系、人才培养模式提出了挑战,所以应该尽早地把这种设计的技术理念引入课堂的教学中,使学生尽早掌握这方面的设计知识,为以后研发打下基础。目前,虽然有较多的FPGA、CPLD方面的书籍,但入门的基础性用书却不是很多,为此,编者编写了本书,希望对引导学生掌握该设计技术有所帮助。

全书比较详细地介绍了在Quartus Ⅱ 9.0平台上,用Verilog HDL进行数字电路设计的方法和技术。

EDA技术发展迅速,涉及领域不断扩大,加之本人水平有限,书中难免有错误的地方,请广大读者批评指正。

王书志

2011年3月

目 录

第一章 Verilog HDL / 001

1.1 Verilog HDL 概述 / 001

1.2 Verilog HDL 的基本知识 / 003

1.3 Verilog HDL 的模块 / 003

1.4 Verilog HDL 的运算符、数据类型及基本词法 / 005

1.5 Verilog HDL 的描述方式 / 012

1.6 Verilog HDL 的行为语句 / 014

第二章 Quartus Ⅱ 9.0 的基本使用方法 / 025

2.1 概述 / 025

2.2 设计流程 / 025

2.3 基本门电路设计 / 051

第三章 组合逻辑电路设计 / 060

3.1 编码器 / 060

3.2 加法器 / 063

3.3 数据选择器 / 068

3.4 数据分配器 / 070

3.5 译码器 / 072

3.6 数据比较器 / 079

第四章 触发器设计 / 084

4.1 电平触发的 RS 触发器 / 091

4.2 D 触发器 / 093

4.3 JK 触发器 / 095

第五章 时序逻辑电路设计 / 101

5.1 异步二进制加法计数器 / 105

5.2 同步二进制加法计数器(2,4,8,16 分频器) / 108

5.3 寄存器 / 110
5.4 锁存器 / 114
5.5 二十九进制计数器 / 117
5.6 一百进制计数器 / 119
5.7 序列信号发生器 / 122
第六章 综合逻辑电路设计 / 125
6.1 函数发生器 / 125
6.2 照明控制电路 / 129
6.3 正弦波发生器 / 131
6.4 简易数字电子钟 / 135
6.5 多波形发生器 / 138
附录 / 154
参考文献 / 161

第一章 Verilog HDL

1.1 Verilog HDL 概述

随着电子设计技术的快速发展,产品设计的集成度和复杂度不断提高,传统的设计方法已经不能满足这样的设计的要求。借助 EDA 工具,使用一种描述语言,对数字电路和数字逻辑系统进行形式化的描述,从而进行电子设计,这种描述语言就是硬件描述语言(HDL, Hardware Description Language)。经过二十多年的发展,硬件描述语言在国外有很多种,这些硬件描述语言有的是从 C 语言发展起来的,有的是从 Pascal 发展起来的,目前在我国使用的硬件描述语言有 VHDL、Verilog HDL 和 AHDL,前两种语言经过发展已经成为国际电气与电子工程师协会(IEEE,International Electrical&Electronic Engineer)标准,应用最广泛。

Verilog HDL 是一种广泛使用的硬件描述语言,用于从算法级、门级到开关级的多种抽象设计层次的数字系统建模。被建模的数字系统对象的复杂性可以介于简单的门和完整的电子数字系统之间。数字系统能够按层次描述,并可在相同描述中显式地进行时序建模。Verilog HDL 可以用于描述:设计的行为特性;设计的数据流特性;设计的结构组成;包含响应监控和设计验证方面的时延和波形产生机制。所有这些都使用同一种建模语言。此外,Verilog HDL 还提供了编程语言接口,通过该接口可以在模拟、验证期间从设计外部访问设计,包括模拟的具体控制和运行。Verilog HDL 不仅定义了语法,而且对每个语法结构都定义了清晰的模拟、仿真语义。因此,用这种语言编写的模型能够使用 Verilog 仿真器进行验证。Verilog HDL 从 C 语言中继承了多种操作符和结构。Verilog HDL 提供了扩展的建模能力,其中许多扩展最初很难理解。但是,Verilog HDL 的核心子集非常易于学习和使用,这对大多数建模应用来说已经足够。当然,完整的硬件描述语言足以对从最复杂的芯片到完整的电子系统进行描述。

1.1.1 Verilog HDL 的发展历史

Verilog HDL 最初是于 1983 年由 Gateway Design Automation 公司为其模拟器产品开发的硬件语言,是由 Gateway Design Automation 公司的 Phile Moorby 首创的。当时只是一种专用语言。该公司的模拟、仿真器产品的广泛使用,使 Verilog HDL 作为一种便于使用且实用的语言逐渐为众多设计者所接受。Verilog HDL 在 1990 年被推向公众领域。Verilog HDL 于

1995 年成为 IEEE 标准,称为 IEEE Std 1364—1995 标准。2001 年,在原标准的基础上经过改进、补充,IEEE 发布了 IEEE Std 1364—2001 标准。

1.1.2 Verilog HDL 的主要能力

(1)基本逻辑门(例如 and、or 和 nand 等)都内置在语言中。

(2)具有用户定义原语(UDP)创建的灵活性。用户定义的原语既可以是组合逻辑原语,也可以是时序逻辑原语。

(3)开关级基本结构模型(例如 pmos 和 nmos 等)也被内置在语言中。

(4)提供显式语言结构,指定设计中的端口到端口的时延及路径时延和设计的时序检查。

(5)可采用三种不同方式或混合方式对设计建模。这些方式包括:行为描述方式——使用过程化结构建模;数据流方式——使用连续赋值语句方式建模;结构化方式——使用门和模块实例语句描述建模。

(6)Verilog HDL 中有两个数据类型:连线(wire)型和寄存器(reg)型。连线型表示构件间的物理连线,而寄存器型表示抽象的数据存储元件。

(7)能够描述层次设计,可使用模块实例结构描述任何层次。

(8)设计的规模可以是任意的,Verilog HDL 不对设计的规模(大小)施加任何限制。

(9)不再是某些公司的专有语言而是 IEEE 标准。

(10)人和机器都可阅读 Verilog HDL,因此它可作为 EDA 的工具和设计者之间的交互语言。

(11)Verilog HDL 的描述能力能够通过使用编程语言接口(PLI)机制进一步扩展。PLI 是允许外部函数访问 Verilog 模块内信息、允许设计者与模拟器交互的例程集合。

(12)设计能够在多个层次上加以描述,从开关级、门级、寄存器传送级(RTL)到算法级,包括进程和队列级。

(13)能够使用内置开关级原语在开关级对设计完整建模。

(14)同一语言可用于生成模拟激励和指定测试的验证约束条件,例如输入值的指定。

(15)能够监控模拟验证的执行,即模拟验证执行过程中设计的值能够被监控和显示。这些值也能够用于与期望值比较,在不匹配的情况下,打印报告消息。

(16)在行为级描述中,Verilog HDL 不仅能够在 RTL 级上进行设计描述,而且能够在体系结构级进行设计描述及在其算法级行为上进行设计描述。

(17)能够使用门和模块实例化语句在结构级进行结构描述。

(18)还具有内置逻辑函数,例如 &(按位与)和|(按位或)。

(19)对于高级编程语言结构,例如条件语句、情况语句和循环语句,语言中都可以使用。

(20)可以显式地对并发和定时进行建模。

(21)提供强有力的文件读写能力。

(22)语言在特定情况下是非确定性的,即在不同的模拟器上模型可以产生不同的结果。

1.2 Verilog HDL的基本知识

1.2.1 Verilog HDL与VHDL比较

这两种语言都是用于数字系统设计开发的硬件描述语言,而且都已经成为IEEE标准。VHDL是美国军方组织开发的HDL,1987年就成为IEEE标准;而Verilog HDL是由一家公司开发的,1995年才成为IEEE标准。Verilog HDL的优势是它很容易掌握,是类C语言,只要有C语言的基础,经过较短时间的学习加上一些实际的操作训练过程,可以在两到三个月内掌握。而掌握VHDL相对就要难一点,因为VHDL是类Ada语言,不太直观,需要有Ada编程基础,一般认为需要至少半年的专业培训才能掌握。目前在美国、日本应用Verilog HDL和VHDL的比率大约是8:2;而在欧洲VHDL应用相对比较多;在我国,很多集成电路设计公司采用的是Verilog HDL。

1.2.2 Verilog HDL与C语言比较

Verilog HDL是在C语言的基础上发展起来的,仍然保留了C语言的结构和特点,两者语法相似。因此,有C语言基础,就可以很快掌握Verilog HDL。

Verilog HDL与C语言有以下方面的区别:

(1)C语言由函数组成,而Verilog HDL由模块(module)组成;

(2)C语言通过函数名及其端口变量实现调用,而Verilog HDL由模块名和端口变量实现调用;

(3)C语言有主函数main(),按顺序执行,而Verilog HDL的各个module是等价的,需有一个顶层模块,包含了芯片系统与外界的所有I/O信号,所有的module并发执行。

1.3 Verilog HDL的模块

1.3.1 什么是模块

模块是Verilog HDL的基本描述单位,用于描述设计的功能或者结构及其与其他模块通信的外部端口。结构可使用开关级原语、门级原语和用户定义的原语方式描述;数据流行为使用连续赋值语句进行描述;时序行为使用过程结构描述。一个模块可以在另一个模块中使用。

以下为使用连续赋值语句进行描述简单的门电路:

```
module and_2(A,B,F);
input A,B;
```

```
output F;
assign F=A&B;
endmodule
```

以上是 Verilog HDL 对简单的门电路——与门的描述,可以看出,程序处在关键字 module 和 endmodule 之间,第一行 module 后为模块名,括号内是端口;第二行是端口定义,定义 A,B 为输入端口;第三行也是端口定义,定义 F 为输出端口;第四行为数据流描述,即连续赋值语句;第五行为关键字。在数字电路中进行数字系统设计时,传统的设计方法是采用自底向上的设计方法,随着电路复杂程度、芯片集成度的提高,该方法已经不能满足实际要求,从而采用自顶向下的设计方法,这样可以进行任务的分配,实现分层管理。图 1.1 是该方法的设计结构图:

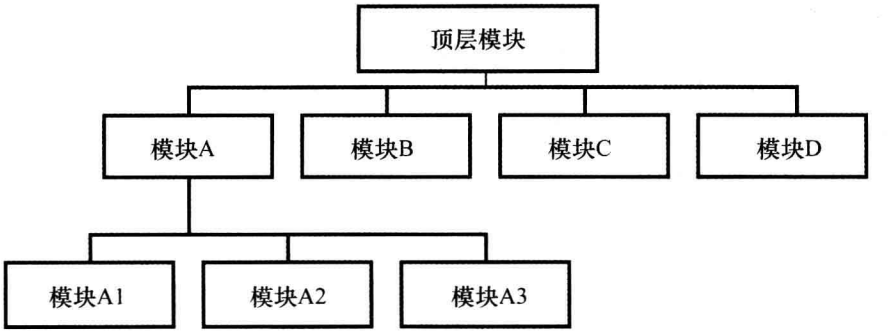

图 1.1 自顶向下的设计结构图

1.3.2 时延

Verilog HDL 中的时延都是根据时间单位来定义的。用时间尺度命令来说明时间单位以及时间精度,其格式为:

`timescale<时间单位><时间精度>

其中的两个参数数字必须是整数,单位为 s,ms,us(10^{-6} s),ns,ps,fs。如 `timescale 1ns / 1ns。此语句为编译器指令,编译器指令"`timescale"将模块中所有时延的单位设置为 1 ns,时间精度为 1 ns(时间精度是指所有的时延必须被限定在 1 ns 内)。

例如,以下为带有时延的连续赋值语句:

```
assign #2 S=A^B;   //#2 指 2 个时间单位,即代表 2 ns。
```

1.3.3 Verilog 模块的模板

模块(module)是 Verilog HDL 语言中最基本的单位,编写模块程序是十分重要的,以下给出的是编写模块程序的模板,开始编写 Verilog HDL 程序时可以参照该模板。

```
module  <顶层模块名>(<输入输出端口列表>);
input   输入端口列表;  //输入端口声明
output  输出端口列表;  //输出端口声明
                       //信号数据类型声明,如果该处没有定义,综合器默认为连线
                         型;
```

```
reg  信号名;  //声明信号为寄存器型
wire  信号列表;  //声明信号为连线型
//逻辑功能描述:
assign <结果信号名>=<表达式>;  //使用 assign 语句描述逻辑功能
always @(<敏感信号表达式>)  //用 always 块描述逻辑功能:
    begin
        //过程赋值
        //if-else,case 语句
        // forever,while,repeat,for 循环语句
        //task,function 调用
    end
//调用:
//调用其他模块:
<调用模块名 module_name > <例化模块名> (<端口列表>);
//门元件:
门元件关键字 <例化门元件名> (<端口列表>);
endmodule
```

1.4 Verilog HDL 的运算符、数据类型及基本词法

1.4.1 Verilog HDL 的运算符

运算符是完成一定运算的符号,Verilog HDL 提供了丰富的运算符。运算符按照在表达式中的运算关系分为单元运算符、双元运算符和三元运算符。单元运算符就是只有一个运算对象,双元运算符有两个运算对象,三元运算符有三个运算对象。运算符按照功能划分又分为算术运算符、逻辑运算符、位运算符、关系运算符、等式运算符、移位运算符、条件运算符和并接运算符等。

(一)算术运算符

1."+":加法运算符或者正运算符,如 a+b,+8

2."-":减法运算符或者负运算符,如 a-b,-8

3."*": 乘法运算符,如 a*b

4."/": 除法运算符,如 3/7

5."%":取模运算符或称为求余运算符,要求%两侧数据为整数型数据

注意:

(1)在进行算术运算时,如果操作数有一个为不确定值 x,则结果也是 x;

(2)在进行除法运算时,结果舍去小数,只取整数;在进行区模运算中,结果的符号位与第一个操作数的符号位相同。如:13%3 结果为 1,15%5 结果为 0,-10%3 结果为-1。

二、逻辑运算符

1.“&&”:逻辑与运算符

两个进行与的操作数都为真时,结果为真;有一个为假时,结果为假。

2.“||”: 逻辑或运算符

两个运算的操作数有一个为真时,结果为真;只有都不为真时,结果才为假。

3.“! ”:逻辑非运算符,是把逻辑运算结果取反

如果操作数值为真,经过该运算,结果就为假;如果操作数值为假,经过该运算结果就为真。

表 1.1 三种逻辑运算的真值表

a	b	! a	! b	a&&b	a||b
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

三、位运算符

位运算符是对其操作数的每一位进行相应的运算。

1.“~”:位操作的非,即对操作数的每一位取反,如~{0,1,1,0}=1001

2.“&”:位操作的与,即对操作数的每一位相与

3.“|”:位操作的或,即对操作数的每一位相或

4.“^”:位操作的异或,即对操作数的每一位相异或

5.“^~”:位操作的同或,即对操作数的每一位相同或

例如:a=3b101;b=3b010

~a=3b010

a&b=3b000

a|b=3b111

a^b=3b111

a^~b=000

注意:

(1)如果操作数长度不相等,较短的操作数将用 0 来补位逐位运算,返回值是一个与两个操作数中位宽较大的一个等宽的值。

(2)不要将逻辑运算符和位运算符相混淆,如“! ”是逻辑非而“~”是位操作的非,也就是按位取反。

四、关系运算符

关系运算符用于比较变量或者常量之间的关系, 在进行关系运算时如果操作数之间的关系成立,则返回值为 1,如果关系不成立,则返回值为 0,如果某一个操作数的值是不确定,则关系是不确定的,则返回不定值 x。

1.“>”:大于

2.“<”:小于

3.“>=”:大于等于

4.“<=”:小于等于

5.“= =”:逻辑相等

6.“! =”:逻辑不相等

7.“= = =”:实例相等

8.“! = =”:实例不相等

注意:

“= =”参与运算的操作数必须是每一位相等,则返回值才为 1,如果参与运算的操作数个别位有高阻态或者不定值,则返回值为不定值;而“= = =”对出现高阻或者不定的位也进行比较,如果操作数完全相同,则返回值为 1,有不同的则返回值为 0。如 a=3b1x10,b=1x10,“a= =b”返回值为 x,而“a= = =b”的返回值为 1。

以下是关系运算符的实例:

```
module Comparison(a,b,out1,out2,out3,out4)
inout [2..0]a,b;
output out2,out3,out4;
reg out2,out3,out4;
always@(a or b)
begin
out1=a<b;
out2=a<=b;
out3=a>b;
if (a>=b);
out4=1
else
out4=0
end
endmodule
```

五、移位运算符

1.“>>”:右移位运算符

2.“<<”:左移位运算符

要移动的位数由第二个操作数来决定, 如 a=4`b1001>>1=4`b0100,a=4`b1001>>4=4`b0000,a=4`b1001<<1=5`b10010。在 Verilog HDL1364—1995 中只有逻辑移位操作,Verilog HDL1364—2001 增加了算术移位操作。例如 D=8'b10100011,则:D >> 3,经过逻辑移位操作结果为:8`b00010100;经过算术移位操作结果为:8`b11110100。

以下为移位操作具体的例子:

```
module shift
reg[3..0] a,b
initial
```

```
begin
a=2;   //a=0010
b=(a<<2);   //b=1000
end
endmodule
```

六、一元缩位运算符

一元缩位运算符是一元运算符,其运算规则与位运算相同,不同的是运算过程不一样,位运算是对操作数的位进行相应的逻辑运算,参加运算的操作数是几位,运算结果也是几位;而一元缩位运算符是对一个操作数进行的运算,最后结果是一位二进制数,即先把操作数的第一位与第二位进行运算,再将上面的运算结果与第三位进行运算,然后再把以上结果与第四位进行运算,以此类推,最后得到一位二进制数。

1."&":与

2."~&":与非

3."|":或

4."~|":或非

5."^":异或

6."^~":同或

如:reg[3:0] a;

```
b=|a;   //b=((a[0]|a[1])|a[2])|a[3];
```

七、条件运算符

"?:":条件运算符

要有三个操作数,即三元运算,通过它把三个表达式连接成一个条件表达式,其格式如下:

逻辑表达式?:操作数 1:操作数 2

如果第一个表达式为真,算子返回第二个操作数,即操作数 1;如果一个表达式为假,算子返回第三个操作数,即操作数 2。常用该运算符来实现数据选择器。如,对于二选一数据选择器,可以用该运算符进行描述:

```
F=al? B:A;//当 al=1 时,F 等于 B,al=0 时,F=A;
```

八、并接运算符

"{}": 并接运算符

这是 Verilog HDL 中一个特殊的运算符,这一运算符可以将两个或更多个信号的某些位并接起来进行运算操作,即把某些信号的某几位详细列出来,中间用逗号分开,最后用大括号括起来,表示一个整体信号。

其格式为:{信号 1 的某几位,信号 2 的某几位,信号 3 的某几位……信号 n 的某几位},如:{a,b[2:0],c,3b010},也可以写成:{a,b[2],b[1],b[0],c,1b'0,1b'1,1b'0}。

并接运算还可以用简化表达式表示,如:

{3{a}},相当于{a,a,a}

运算符的优先级顺序如表 1.2 所示。

表 1.2 运算符的优先级别表

级别序号	运算符	优先级别
1	! ~	最高级别
2	* / %	
3	+ -	
4	<< >>	
5	<<= >>=	
6	== ! = === ! ==	
7	&	
8	^ ^~	
9	|	
10	&&	
11	||	
12	?:	最低级别

1.4.2 Verilog HDL 的数据类型

Verilog HDL 中总共有两种数据类型：一种是连线型；另外一种是寄存器型。在这两类中最常见的是以下四种数据类型：integer 型、parameter 型、reg 型和 wire 型。除此之外，还有一些其它的数据类型：large 型、medium 型、scalared 型、time 型、small 型、tri 型、trio 型、tril 型、triand 型、trior 型、trireg 型、vectored 型、wand 型和 wor 型，共 18 种数据类型。

一、常量

Verilog HDL 的数值集合由下面四种基本的值组成。

1.“0”：逻辑 0，低电平，或者假

2.“1”：逻辑 1，高电平，或者真

3.“x”：逻辑不定状态

4.“z”：高阻态

二、数字

数字按其数值类型分为整数(integer)和实数(real)。

1.整数

有二进制、十进制、八进制、十六进制，书写格式如下：

+/-<位宽>'<基数><数字>

“+/-”：为正负号，负数用二进制补码给出。

位宽：描述常量所含的位数的十进制数，可以选择，如果没有这一项，可以由常量的值推出。

基数：b、B 为二进制数；d、D 为十进制数；o、O 为八进制数；h、H 为十六进制数。基数缺省默认为十进制数。

数字：由基数决定的一串 ASCAL 字符，如果基数为 b、B，值可以是 0，1，x，X，z，Z；如果基

数为 d、D,值可以是 0~9; 如果基数为 o、O,值可以是 0~7;如果基数为 h、H,值可以是 0~9,a,b,c,d,e,f(A,B,C,D,E,F)。

如:

```
8'b1010 1111      //位宽为 8 位的二进制数
16                //十进制数 16
8'h16             //十六进制数 16
8'h6x             //位宽为 8 的十六进制数,其低 4 位值为不定值
4'b100z           //位宽为 4 的二进制数,从低位数起第一位为高阻值
12'dz             //位宽为 12 的十进制数,其值为高阻值
```

注意:

(1)数值常量中的下画线是为了增强可读性,可以忽略;

(2)当没有位宽数字时,默认是 32 位;

(3)x、z 在二进制数中代表一位,在八进制数中代表三位,在十六进制数中代表四位;

(4)数值常量中的"?"表示高阻态,如 2'b0?代表一个两位的二进制数,其中的一位是高阻态。

2.实数

Verilog HDL 中实数既可以用小数表示,如 0.8,1.2,也可以用科学计数法的方式表示,如 17e6,即 17 乘 10 的 6 次方。实数可以转化为整数,转化的原则是四舍五入法则,如 18.6 转化为整数为 19,而 18.2 转化为整数则为 18。

三、字符串

字符串常量是写在双引号之间的字符序列串,如"HELLO";字符串变量是寄存器型变量,它有与字符串的字符数乘以 8 相等的位宽,如"HELLO",是一个 5 个字符的字符串,需要 8*5=40 即 40 宽的寄存器。

四、参数(parameter)型

用"parameter"定义一个标识符,它代表了一个常量,称为符号常量。如:parameter lik=4'b1010,定义参数 lik 代表常量(四位二进制数)。

1.4.3 Verilog HDL 的基本词法

Verilog HDL 是由各种符号构成的,这些符号有:标识符、注释、空白符、数据类型、运算符。

一、标识符

标识符是用来标识程序中变量、语句、数据类型以及函数等的名字的,它可以是一组字母、数字下画线等的组合,但是第一个字母必须是字母或者是下画线。标识符区分大小写。

二、关键字

关键字是 Verilog HDL 内部专用的词,是编程语言保留的特殊标识符。它有特定的、专用的语法作用,用户不能对其做新的定义,它区分大小写,只有小写才是关键字,大写不是关键字。IEEE 标准 Verilog HDL1364—1995 中规定了 102 个关键字,在 Verilog HDL1364—2001 中规定了 123 个关键字。IEEE 标准 Verilog HDL1364—2001 中规定的关键字如表 1.3 所示。