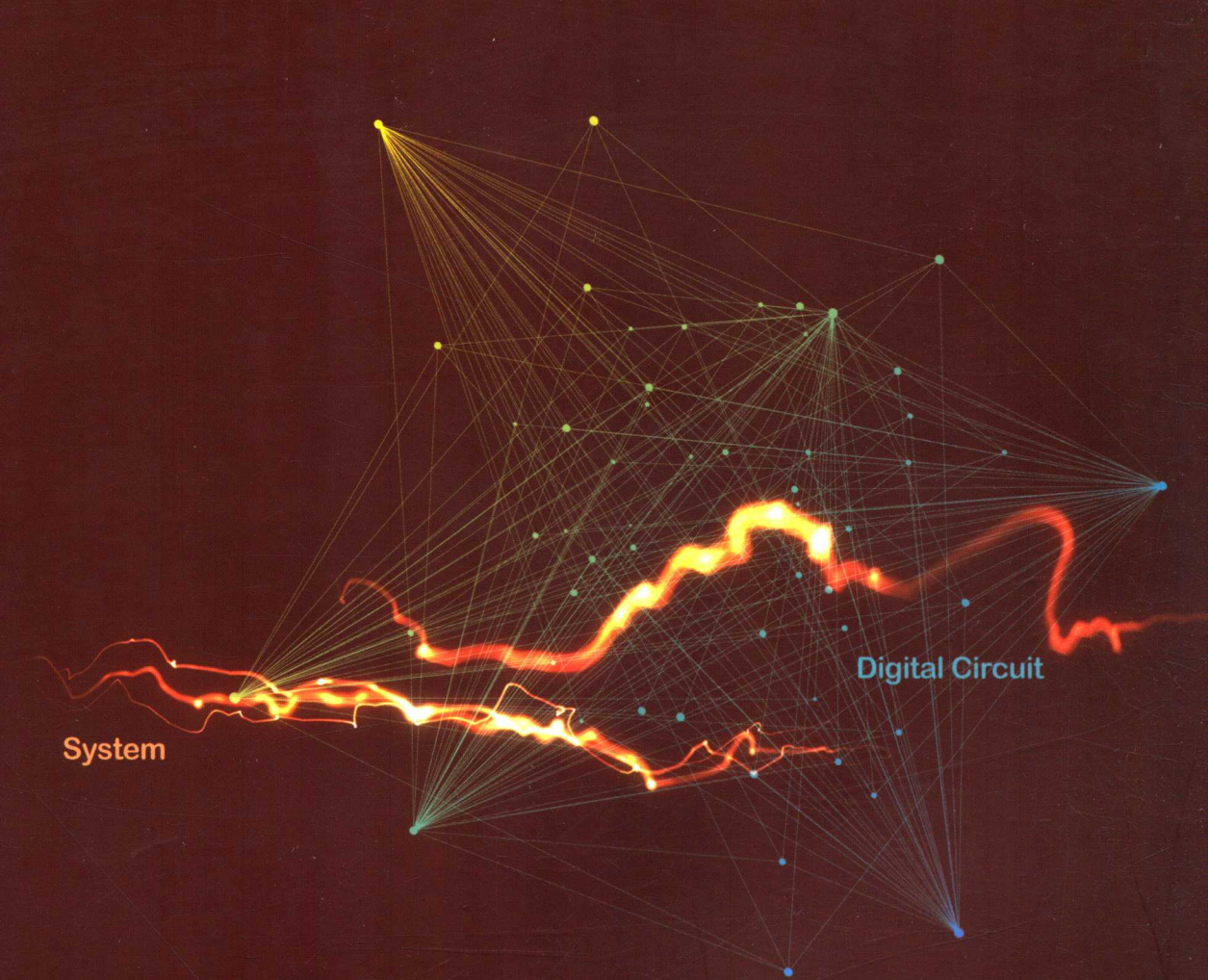

现代数字电路与系统设计

◎ 江国强　编著

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

现代数字电路与系统设计

江国强　编著

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

内 容 简 介

本书是基于电子设计自动化（EDA）技术编写的，全书共 8 章，包括 Verilog HDL、门电路的设计、组合逻辑电路的设计、触发器的设计、时序逻辑电路的设计、存储器的设计、数字系统的设计和常用 EDA 软件。数字电路与系统设计都是基于 Verilog HDL 完成的，每个设计都经过了 EDA 软件的编译和仿真，或经过 EDA 实验开发系统平台的验证，确保无误。

本书图文并茂、通俗易懂，可作为高等学校工科相关专业数字逻辑电路、EDA 技术与应用、可编程逻辑器件等课程的教学参考书，或课程设计、实训和毕业设计的参考书，也可作为从事数字电路与系统设计的工程技术人员的参考书。

图书在版编目（CIP）数据

现代数字电路与系统设计/江国强编著. —北京：电子工业出版社，2017.7
ISBN 978-7-121-31571-8
Ⅰ. ①现… Ⅱ. ①江… Ⅲ. ①数字电路－系统设计 Ⅳ. ①TN79

中国版本图书馆 CIP 数据核字（2017）第 107957 号

责任编辑：韩同平　　特约编辑：邹凤麒　王博　段丹辉
印　　刷：北京京师印务有限公司
装　　订：北京京师印务有限公司
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：787×1092　1/16　印张：16　　字数：512 千字
版　　次：2017 年 7 月第 1 版
印　　次：2017 年 7 月第 1 次印刷
定　　价：49.90 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：hantp@phei.com.cn。

前　言

在 20 世纪 90 年代，国际上电子和计算机技术先进的国家，一直在积极探索新的电子电路设计方法和设计工具，并取得巨大成功。在电子设计技术领域，可编程逻辑器件 PLD（Programmable Logic Device）的应用，已得到很好的普及，它为数字系统的设计带来极大的灵活性。该器件可以通过软件编程而对其硬件结构和工作方式进行重构，使得硬件的设计可以如同软件设计那样方便快捷，极大地改变了传统的数字系统设计方法、设计过程和设计观念。随着可编程逻辑器件集成规模不断扩大、自身功能不断完善，以及计算机辅助设计技术的提高，使现代电子系统设计领域的电子设计自动化 EDA（Electronic Design Automation）技术应运而生。传统的数字电路设计模式，如利用卡诺图的逻辑化简手段、布尔方程表达式设计方法和相应的中小规模集成电路的堆砌技术正在迅速地退出历史舞台。

本书是基于硬件描述语言 HDL（Hardware Description Language）编写的。目前，国际最流行的、并成为（美国）电机及电子工程师学会（IEEE，Institute of Electrical and Electronics Engineers）标准的两种硬件描述语言是 VHDL 和 Verilog HDL，两种 HDL 各具特色。但 Verilog HDL 是在 C 语言的基础上演化而来的，只要具有 C 语言的编程基础，就很容易学会并掌握这种语言，而且国内外 90%的电子公司都把 Verilog HDL 作为企业标准设计语言，因此本书以 Verilog HDL 作为数字电路与系统的设计工具。

本书共 8 章，首先介绍 Verilog HDL，然后介绍基于 Verilog HDL 的常用数字电路和一些专用数字电路的设计。所谓常用数字电路是指用途比较广泛并形成集成电路产品的电路，例如 TTL 系列和 CMOS 系列的集成电路产品。专用数字电路是指具有特定功能的电路，例如序列序号发生器、序列序号检测器等，但它们没有现成的集成电路产品。另外还介绍了一些通俗易懂的数字系统设计和一些常用的 EDA 软件。

第 1 章 Verilog HDL，介绍 Verilog HDL 的语法规则、语句和仿真方法，为基于 Verilog HDL 的数字电路及系统的设计打下基础。

第 2 章门电路的设计，介绍普通门、三态输出门和三态驱动门的设计。

第 3 章组合逻辑电路的设计，介绍算术运算电路、编码器、译码器、数据选择器、数据比较器、奇偶校验器和码转换器等组合逻辑电路的设计。

第 4 章触发器的设计，介绍基本 RS 触发器、钟控 RS 触发器、D 触发器和 JK 触发器的设计。

第 5 章时序逻辑电路的设计，介绍数码寄存器、移位寄存器和计数器等常用时序逻辑电路的设计，还介绍顺序脉冲发生器、序列序号发生器，伪随机信号发生器、序列序号检测器、码转换器和串行数据检测器等专用数字电路的设计。

第 6 章存储器的设计，介绍只读存储器 ROM 和随机存储器 RAM 的设计。

第 7 章数字系统设计，首先介绍数字系统的设计方法，然后介绍串行加法器、24 小时计时器、万年历、倒计时器、交通灯控制器、出租车计费器、波形发生器、数字电压表和数字频率计等系统电路的设计。

第 8 章常用 EDA 软件，介绍 Quartus II 13.0、ModelSim、Matlab/DSP Builder 和 Nios II 等常用的 EDA 软件，供读者在数字电路及系统设计时参考。

本书中的所有 Verilog HDL 程序都经过美国 Altera 公司的 Quartus II 软件的编译和仿真，或经过 EDA 实验开发系统平台验证，确保无误。为了使读者看清楚仿真结果，大部分设计的仿真结果用 Quartus II 9.0 版本软件中的自带仿真工具（Waveform Editor）或 Quartus II 13.0 版本软件中的大学计划仿真工具（university program vwf）实现的。

本书由桂林电子科技大学江国强教授编著，如有不足之处，恳请读者指正。

E-mail：hmjgq@guet.edu.cn

地 址：桂林电子科技大学（541004）

电 话：（0773）5601095，13977393225

编著者

目　录

第 1 章　Verilog HDL

Verilog HDL 是目前应用最为广泛的硬件描述语言，并被 IEEE 采纳为 IEEE 1364-1995 标准（Verilog-1995 版本），2001 年升级为 Verilog-2001 版本，2005 年升级为 System Verilog-2005 版本。目前，在大部分 EDA 软件中，Verilog-2001 是默认版本。Verilog HDL 可以用来进行各种层次的逻辑设计，也可以进行数字系统的逻辑综合、仿真验证和时序分析。Verilog HDL 适合算法级（Algorithm）、寄存器传输级（RTL）、逻辑级（Logic）、门级（Gate）和版图级（Layout）等各个层次的电路设计和描述。

本章介绍 Verilog HDL 的语法规则、语句和仿真方法，为基于 Verilog HDL 的数字电路与系统的设计打下基础。

1.1　Verilog HDL 设计模块的基本结构

Verilog HDL 程序设计是由模块（module）构成的，设计模块的基本结构如图 1.1 所示。一个完整的 Verilog HDL 设计模块包括模块端口定义、I/O 声明、信号类型声明和功能描述 4 个部分。

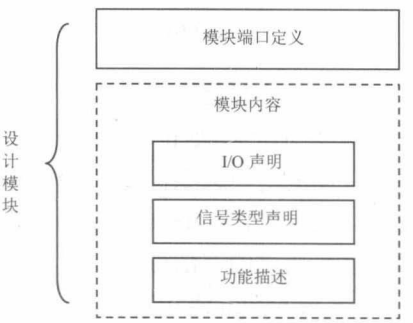

图 1.1　Verilog HDL 程序模块结构图

1．模块端口定义

模块端口定义用来声明电路设计模块名称及相应的输入和输出端口，端口定义格式如下：

```
module 模块名(端口 1, 端口 2, 端口 3, …);
```

模块名是设计电路的名称，它由用户按照标识符规则命名，它也是保存的源文件名称。例如，设计一个 2 输入 4 与非门电路时，可以用 CT7400 作为模块名，并用 CT7400.v（.v 是 Verilog HDL 源文件的属性后缀）保存设计的源文件。在端口定义的圆括弧中，是设计电路模块与外界联系的全部输入和输出端口名称或引脚，它是设计模块对外的一个通信界面，是外界可以看到的部分（不包含电源和接地端），多个端口名之间用“,”分隔。例如，用 adder1 作为 1 位全加器的 Verilog HDL 设计模块名，sum 是求和输出，cout 是向高位的进位输出，a 和 b 是两个加数的输入，cin 是低位进位输入，则 adder1 模块的端口定义为：

```
module adder1(sum,cout,a,b,cin);
```

2．模块内容

模块内容包括 I/O 声明、信号类型声明和功能描述。

（1）模块的 I/O 声明

模块的 I/O 声明用来声明模块端口定义中各端口数据流动方向，包括输入（input）、输

出（output）和双向（inout）。I/O 声明格式如下：

```
input     端口 1, 端口 2, 端口 3, …;     //声明输入端口
output    端口 1, 端口 2, 端口 3, …;     //声明输出端口
```

例如：

```
input     a,b,cin;
output    sum,cout;
```

（2）变量类型声明

变量类型声明用来声明设计电路中所用的变量的数据类型。变量的类型主要有连线（wire）、寄存器（reg）、整型（integer）、实型（real）和时间（time）等。例如：

```
wire      a,b,cin;          //声明 a,b,cin 是 wire 变量
reg       cout;             //声明 cout 是 reg 型变量
reg [7:0] q;                //声明 q 是 8 位 reg 型变量
```

在 Verilog HDL 的 2001 版本或以上版本，允许将 I/O 声明和变量类型放在一条语句中，例如：

```
output reg[7:0]  q;    //声明 q 是 8 位 reg 型输出变量
```

（3）功能描述

功能描述是 Verilog HDL 程序设计中最主要的部分，用来描述设计模块的内部结构和模块端口间的逻辑关系，在电路上相当于器件的功能或内部电路结构。功能描述可以用 assign 语句、元件例化（instantiate）、always 块语句和 initial 块语句方法来实现，通常把确定这些设计模块描述的方法称为建模。

① 用 assign 语句建模

用 assign 语句建模的方法很简单，只需要在 assign 后面加一个表达式即可。assign 语句一般适合对组合逻辑进行赋值，称为连续赋值方式。用 assign 赋值的语句如下：

```
assign    {co,sum} = a+b+cin;
```

② 用元件例化（instantiate）方式建模

元件例化方式建模是利用 Verilog HDL 提供的元件库实现的。元件库包括与、或、非、与非、或非、异或等基本元件和一些特殊元件。例如，用与门元件例化一个 3 输入端与门的语句为：

```
and  u1(y,a,b,c);
```

其中，and 是 Verilog HDL 元件库中与门元件名，u1 是可选的例化名，它相当于元件 and 插在印刷电路板上的插座名，y 是与门的输出端，a、b 和 c 是输入端。

③ 用 always 块语句建模

always 块语句可以用于设计组合逻辑和时序逻辑，但时序逻辑必须用 always 块语句来建模。一个程序设计模块中，可以包含一个或多个 always 块语句。程序执行中，若 always 块语句的敏感参数发生变化，就执行一遍 always 块中的语句，产生新的结果。定义 always 块的格式为：

```
always    @(posedge clk or negedge clrn)
```

其中，“@(posedge clk or negedge clrn)”敏感参数表，说明always块中的语句只有在clk的上升沿或clrn的下降沿到来时才能执行。

④ 用initial块语句建模

initial块语句与always语句类似，不过在程序中它只执行1次就结束了，initial块语句常用于设计电路的初始化操作和仿真，例如在电子日历的设计中，可以将日历的日期初始化为2000（年）01（月）01（日）。

Verilog HDL程序设计模块的基本结构归纳如下：

1）Verilog HDL程序是由模块构成的。每个模块的内容都是嵌在module和endmodule两语句之间，每个模块实现特定的功能，模块是可以进行层次嵌套的。

2）每个模块首先要进行端口定义，并声明端口属于输入（input）、输出（output）或双向（inout），然后对模块的功能进行逻辑描述。

3）Verilog HDL程序的书写格式自由，一行可以一条或多条语句，一条语句也可以分为多行写。

4）除了end或以end开头的关键字（如endmodule）语句外，每条语句后必须要有分号“;”。程序中的关键字全部用小写字母书写，标点符号全部用半角符号。

1.2 Verilog HDL的词法

Verilog HDL源程序由空白符号分隔的词法符号流所组成。词法符号包括空白符、注释、操作符、常数、字符串、关键词和标识符。准确无误地理解和掌握Verilog HDL的词法的规则和用法，对正确地完成Verilog HDL源程序的设计十分重要。

1．空白符和注释

Verilog HDL的空白符包括空格、tab符号、换行和换页。空白符用来分隔各种不同的词法符号，合理地使用空白符可以使源程序具有一定的可读性和编程风格。空白符如果不是出现在字符串中，编译源程序时将被忽略。

注释分为行注释和块注释两种方式。行注释用符号//（两个斜杠）开始，注释到本行结束。块注释用/*开始，用*/结束。块注释可以跨越多行，但它们不能嵌套。

2．常数

在Verilog HDL中的常数包括数字、未知x和高阻z三种。数字可以用二进制、十进制、八进制和十六进制等4种不同数制来表示，完整的数字格式为：

```
<位宽>'<进制符号><数字>
```

其中，位宽表示数字对应的二进制数的位数宽度；进制符号包括b或B（表示二进制数），d或D（表示十进制数），h或H（表示十六进制数），o或O（表示八进制数）。例如：

```
8'b10110001    //表示位宽为8位的二进制数10110001
8'hf5          //表示位宽为8位的十六进制数f5
```

数字的位宽可以缺省，例如：

```
'b10110001          //表示二进制数
'hf5                //表示十六进制数
```

十进制数的位宽和进制符号可以缺省，例如：

```
125                 //表示十进制数 125
```

另外，用 x（或 X）和 z（或 Z）分别表示未知值和高阻态值，它们可以出现在除十进制数以外的数制形式中。x 和 z 的位数由所在的数制格式决定，在二进制数中，一个 x 或 z 表示 1 位未知位或 1 位高阻位；在十六进制数中，一个 x 或 z 表示 4 位未知位或 4 位高阻位；在八进制数中，一个 x 或 z 表示 3 位未知位或 3 位高阻位。例如：

```
'b1111xxxx      //等价'hfx
'b1101zzzz      //等价'hdz
```

3. 字符串

字符串是用双引号括起来的可打印字符序列，它必须包含在同一行中。例如，“ABC”、“A BOY.”、“A”、“1234”都是字符串。

4. 关键词

关键词（也称关键字）是 Verilog HDL 预先定义的单词或单词的组合，它们在程序中有不同的使用目的。例如，用 module 和 endmodule 来指出源程序模块的开始和结束；用 assign 来描述一个逻辑表达式等。Verilog-1995 版本的关键词有 97 个，Verilog-2001 版本增加了 5 个，共 102 个，如表 1.1 所示。每个关键词全部由小写字母组成，少数关键词中包含“0”或“1”数字。

表 1.1 **Verilog HDL 的关键词**

always	and	assign	begin	buf
bufi0	bufu1	case	casex	casez
cmos	deassign	default	defparam	disable
edge	else	end	endcase	endfunction
endmodule	endprimitive	endspecify	endtable	endtask
event	for	force	forever	fork
function	highz0	highz1	if	initial
inout	input	integer	join	large
macromodule	medium	module	nand	negedge
nmos	nor	not	notif0	nottif1
or	output	pmos	posedge	primitive
pull0	pull1	pulldown	pullup	remos
reg	release	Repeatr	rnmos	rpmos
rtran	rtranif0	reranif1	scalared	small
specify	specparam	strong0	strong1	supply0
supply1	table	task	time	tran

续表

always	and	assign	begin	buf
tranif0	tranif1	tri	tri0	tri1
triand	trior	vectored	wait	wand
weak0	weak1	while	wire	wor
xnot	xor			

5. 标识符

标识符是用户编程时为常量、变量、模块、寄存器、端口、连线、示例和 begin-end 块等元素定义的名称。标识符可以是字母、数字和下画线“_”等符号组成的任意序列。定义标识符时应遵循如下规则：

① 首字符不能是数字。

② 字符数不能多于 1024 个。

③ 大小写字母是不同的（仅限于 Verilog 1995 版本）。

④ 不要与关键词同名。

例如，a、b、adder、adder8、name_adder 都是正确的标识符；而 1a、?b 是错误的标识符。

Verilog HDL 允许使用转义标识符。转义标识符中可以包含任意的可打印字符，转义标识符从空白符号开始，以反斜杠“\”作为开始标记，到下一个空白符号结束，反斜杠不是标识符的一部分。下面是转义标识符的示例：

```
\74LS00
\a+b
```

6. 操作符

操作符也称为运算符，是 Verilog HDL 预定义的函数名字，这些函数对被操作的对象（即操作数）进行规定的运算，得到一个结果。操作符通常由 1～3 个字符组成，例如，“+”表示加操作，“==”（两个=字符）表示逻辑等操作，“===”（3 个=字符）表示全等操作。有些操作符的操作数只有 1 个，称为单目操作；有些操作符的操作数有 2 个，称为双目操作；有些操作符的操作数有 3 个，称为三目操作。

Verilog HDL 的操作符分为算术操作符、逻辑操作符、关系操作符、等值操作符、缩减操作符、条件操作符、位操作符和并接操作符 9 类。

（1）算术操作符（Arithmetic operators）

常用的算术操作符有：+（加）、-（减）、*（乘）、/（除）、%（求余）和**（乘方）6 种。其中%是求余操作符，在两个整数相除的基础上，取出其余数。例如，5 % 6 的值为 5；13 % 5 的值是 3。整除（/）和求余（%）运算符可以方便电路的设计，如将二进制数转换为十进制数（8421BCD 码），但这两种运算符在综合过程中占用很多逻辑单元（LEs），所以一般电路设计最好不要使用。

（2）逻辑操作符（Logical operators）

逻辑操作符包括：&&（逻辑与）、||（逻辑或）、!（逻辑非）。例如，A && B 表示 A 和

B 进行逻辑与运算；A || B 表示 A 和 B 进行逻辑或运算；!A 表示对 A 进行逻辑非运算。

（3）位运算（Bitwise operators）

位运算是将两个操作数按相应位进行逻辑操作。位运算操作符包括：~（按位取反）、&（按位与）、|（按位或）、^（按位异或）、^~或~^（按位同或）。例如，设 A='b11010001，B='b00011001，则：

```
~A = 'b00101110
A & B = 'b00010001
A | B = 'b11011001
A ^ B = 'b11001000
A ^~ B = 'b00110111
```

在进行位运算时，若两个操作数的位宽不同，计算机会自动将两个操作数按右端对齐，位数少的操作数会在高位用 0 补齐。

位运算与逻辑操作符运算的结果是相同的，因此，逻辑操作运算直接可以用位运算替代，例如，A && B 可以写成 A & B。

（4）关系操作符（Relational operators）

关系操作符用来对两个操作数进行比较。关系操作符有：<（小于）、<=（小于等于）、>（大于）、>=（大于等于）。其中，<=也是赋值运算的赋值符号。

关系运算的结果是 1 位逻辑值。在进行关系运算时，如果关系是真，则计算结果为 1；如果关系是假，则计算结果为 0；如果某个操作数的值不定，则计算结果不定（未知 x），表示结果是模糊的。

（5）等值操作符（Equality operators）

等值操作符包括：==（等于）、!=（不等于）、===（全等）、!==（不全等）4 种。

等值运算的结果也是 1 位逻辑值，当运算结果为真时，返回值 1；为假则返回值 0。相等操作符（==）与全等操作符（===）的区别是：当进行相等运算时，两个操作数必须逐位相等，其比较结果的值才为 1（真），如果某些位是不定或高阻状态，其相等比较的结果就会是不定值；而进行全等运算时，对不定或高阻状态位也进行比较，当两个操作数完全一致时，其结果的值才为 1（真），否则结果为 0（假）。

例如，设 A = 'b1101xx01，B = 'b1101xx01，则：

```
A == B    运算的结果为 x（未知）
A === B   运算的结果为 1（真）
```

（6）缩减操作符（Reduction operators）

缩减操作符包括：&（与）、~&（与非）、|（或）、~|（或非）、^（异或）、^~或~^（同或）。缩减操作运算法则与逻辑运算操作相同，但操作的运算对象只有一个。在进行缩减操作运算时，对操作数进行与、与非、或、或非、异或、同或等缩减操作运算，运算结果有 1 位“1”或“0”。例如，设 A = 'b11010001，则& A = 0（在与缩减运算中，只有 A 中的数字全为 1 时，结果才为 1）；|A = 1（在或缩减运算中，只有 A 中的数字全为 0 时，结果才为 0）。缩减操作相当于一个逻辑门，与缩减运算相当于一个与门，只有与门的全部输入为“1”时，输出（1 位）才为“1”，否则输出为“0”。

（7）转移操作符（Shift operators）

转移操作符包括：>>（右移）、<<（左移）。其使用方法为：

```
操作数 >> n;        //将操作数的内容右移 n 位，同时从左边开始用 0 来填补移出的位数。
操作数 << n;        //将操作数的内容左移 n 位，同时从右边开始用 0 来填补移出的位数。
```

例如，设 A = 'b11010001，则 A >> 4 的结果是 A = 'b00001101；而 A << 4 的结果是 A = 'b00010000。

（8）条件操作符(Conditional operators)

条件操作符为：?

条件操作符的操作数有 3 个，其使用格式为：

```
操作数 = 条件 ? 表达式 1: 表达式 2;
```

即当条件为真（条件结果值为 1）时，操作数 = 表达式 1；为假（条件结果值为 0）时，操作数 = 表达式 2。

（9）并接操作符（Concatenation operators）

并接操作符为：{ }

并接操作符的使用格式为：

```
{操作数 1 的某些位, 操作数 2 的某些位, ..., 操作数 n 的某些位};
```

即将操作数 1 的某些位，与操作数 2 的某些位，…，与操作数 n 的某些位并接在一起，构成一个由这些数组成的多位数。例如，将 1 位全加器进位 co 与和 sum 并接在一起使用，它们的结果由两个加数 a、b 及低位进位 cin 相加决定的表达式为：

```
{co,sum}= a+b+cin;
```

（10）操作符的优先级

操作符的优先级如表 1.2 所示。表中顶部的操作符优先级最高，底部的最低，列在同一行的操作符的优先级相同。所有的操作符（?:操作符除外）在表达式中都是从左向右结合的。圆括号可以用来改变优先级，并使运算顺序更清晰，对操作符的优先级不能确定时，最好使用圆括弧来确定表达式的优先顺序，既可以避免出错，也可以增加程序的可读性。

表 1.2　操作符的优先级

优先级序号	操作符	操作符名称	优先级序号	操作符	操作符名称
1	！、~	逻辑非、按位取反	7	&、~&	缩减与、缩减与非
2	* 、/、 %	乘、除、求余	8	^、^~	缩减异或、缩减同或
3	+、-	加、减	9	|、~|	缩减或、缩减或非
4	<<、>>	左移、右移	10	&&	逻辑与
5	<、<=、>、>=	小于、小于等于、大于、大于等于	11	||	逻辑或
6	==、!=、===、!==	等于、不等于、全等、不全等	12	?:	条件操作符

7. Verilog HDL 数据对象

Verilog HDL 数据对象是指用来存放各种类型数据的容器，包括常量和变量。

（1）常量

常量是一个恒定不变的值数，一般在程序前部定义。常量定义格式为：

```
parameter 常量名 1 = 表达式, 常量名 2 = 表达式, …, 常量名 n = 表达式;
```

其中，parameter 是常量定义关键字，常量名是用户定义的标识符，表达式是为常量赋的值。例如：

```
parameter [3:0]   s0 = 'b0001,s1 = 'b0010;
```

上述语句定义了 4 位常量 s0 和 s1，s0 的值为二进制数 0001，s1 的值为二进制数 0010。

（2）变量

变量是在程序运行时其值可以改变的量。在 Verilog HDL 中，变量分为网络型（nets type）和寄存器型（register type）两种。

① 网络型变量（nets type）

nets 型变量是输出值始终根据输入变化而更新的变量，它一般用来定义硬件电路中的各种物理连线。Verilog HDL 提供了多种 nets 型变量，如表 1.3 所示。

在 nets 型变量中，wire 型变量是最常用的一种。wire 型变量常用来表示以 assign 语句赋值的组合逻辑变量。Verilog HDL 模块中的输入和输出变量类型缺省时自动定义为 wire 型。wire 型变量可以作为任何方程式的输入，也可以作为 assign 语句和例化元件的输出。综合而言，wire 型变量的取值可以是 0、1、x 和 z。

表 1.3　常用的 nets 型变量及说明

类　型	功 能 说 明
wire、tri	连线类型（两者功能完全相同）
wor、trior	具有线或特性的连线（两者功能一致）
wand、triand	具有线与特性的连线（两者功能一致）
tri1、tri0	分别为上拉电阻和下拉电阻
supply1、supply0	分别为电源（逻辑 1）和地（逻辑 0）

用 wire 定义的变量有一个范围选项（即位宽），默认的位宽是 1。位宽为 1 位的变量称为标量，位宽超过 1 位的变量称为向量。标量的定义不需要加位宽选项。wire 型变量的定义格式如下：

```
wire 变量名 1, 变量名 2, …, 变量名 n;
```

例如

```
wire        a,b,c;       //定义了 3 个 wire 型的变量，位宽均为 1 位
wire[7:0]   databus;     //定义了 1 个 wire 型的数据总线，位宽为 8 位
wire[15:0] addrbus;      //定义了 1 个 wire 型的地址总线，位宽为 16 位
```

在 Verilog-2001 版本中，wire 变量的定义和 I/O 端口的定义可以写在同一语句中，例如：

```
output wire[7:0] co;    //定义了 1 个位宽为 8 位的 wire 型输出变量
```