*第九章

硬件描述语言简介

内容提要

本章简要介绍硬件描述语言。首先简单介绍了硬件描述语言的基本概念和发展与应用概况，然后介绍了有关 Verilog HDL 的基本知识，最后给出了几个用 Verilog HDL 描述逻辑电路的实例。

9.1 概述

随着半导体技术的发展，数字电路已经由中小规模的集成电路向可编程逻辑器件(PLD)及专用集成电路(ASIC)转变。数字电路的设计手段也发生了变化，由传统的手工方式逐渐转变为以 EDA 工具作为设计平台的工作方式。硬件描述语言(HDL)就是设计人员和 EDA 工具之间的一种界面。利用硬件描述语言并借助 EDA 工具，可以完成从系统、算法、协议的抽象层次对电路进行建模、仿真、性能分析直到 IC 版图或 PCB 版图生成的全部设计工作。

硬件描述语言主要用于编写设计文件，在 EDA 工具中建立电路模型。通过对电路结构或功能行为的描述，可以在不同的抽象层次对电路进行逐层描述，用一系列分层次的模块来表示极其复杂的数字电路系统。

硬件描述语言发展至今已有近 30 年的历史，已经成功地应用于电子电路设计的各个阶段：建模、仿真、验证和综合等。自 20 世纪 80 年代以来，出现了由各个公司自行开发和使用的多种硬件描述语言，这些语言各自面向特定的设计领域和层次。但众多的语言使用户无所适从，也降低了电路设计的可移植性和通用性。因此，急需一种面向设计的多领域、多层次并得到普遍认同的标准硬件描述语言。VHDL 和 Verilog HDL 语言适应了这种趋势的要求，先后被确定为

IEEE 标准。

VHDL 和 Verilog HDL 是目前两种最常用的硬件描述语言。除了这两种最流行的硬件描述语言外,随着系统级 FPGA 以及系统芯片的出现,软硬件协调设计和系统设计变得越来越重要。传统意义上的硬件设计越来越倾向于与系统设计和软件设计结合。为适应新的情况,硬件描述语言也在迅速发展,不断出现新的硬件描述语言,如 Superlog、SystemC、Cynlib C ++ 等。

下面简单介绍用 Verilog HDL 对逻辑电路进行描述的方法。

9.2 Verilog HDL 简介

1983 年 Gateway Design Automation 公司在 C 语言的基础上,为其仿真器产品 Verilog - XL 开发了一种专用硬件描述语言——Verilog HDL。随着 Verilog - XL 成功和广泛的使用,Verilog HDL 被众多数字电路设计者所接受。1989 年,Cadence 公司收购了 GDA 公司。1990 年,Cadence 公司成立了 OVI(Open Verilog International)组织,以促进 Verilog HDL 语言的推广和发展。IEEE 于 1995 年制定了 Verilog HDL 的 IEEE 标准——Verilog HDL 1364 - 1995;2001 年又发布了 Verilog HDL 1364 - 2001 标准,并在其中加入了 Verilog HDL - A 标准,使 Verilog 有了描述模拟电路的能力。

Verilog HDL 从 C 语言中继承了多种操作符和结构,源文本文件由空白符号分隔的词法符号流组成。词法符号的类型有空白符、注释、操作符、数字、字符串、标识符和关键字等,从形式上看和 C 语言有许多相似之处。

9.2.1 基本程序结构

和其他高级语言一样,Verilog HDL 语言采用模块化的结构,以模块集合的形式来描述数字电路系统。模块(module)是 Verilog HDL 语言中描述电路的基本单元。模块对应硬件上的逻辑实体,描述这个实体的功能或结构,以及它与其他模块的接口。它所描述的可以是简单的逻辑门,也可以是功能复杂的系统。模块的基本语法结构如下:

```
module <模块名>( <端口列表> )
 <定义>
 <模块条目>
endmodule
```

根据 <定义> 和 <模块条目> 的描述方法不同,可将模块分成行为描述模块、结构描述模块,或者是二者的组合。行为描述模块通过编程语言定义模块的

状态和功能。结构描述模块将电路表达为具有层次概念的互相连接的子模块，其最底层的元件必须是 Verilog HDL 支持的基元或已定义过的模块。

9.2.2 词法构成

Verilog HDL 的词法标识符包括：间隔符与注释符、操作符、数值常量、字符串、标识符和关键字。

一、间隔符与注释符

间隔符又称空白符，包括空格符、制表符、换行符以及换页符等。它们的作用是分隔其他词法标识符。另外，在必要的地方插入间隔符可以增强源文件的可读性。但在字符串中空格符和制表符是有意义的字符。

Verilog HDL 有单行注释和多行段注释两种注释形式。单行注释以字符“//”起始，到本行结束；而段注释则是以“/*”起始以“*/”结束，在段注释中不允许嵌套，段注释中单行注释标识符“//”没有任何特殊意义。

二、操作符

Verilog HDL 中定义了操作符，又称运算符，按照操作数的个数，可以分为一元、二元和三元操作符；按功能可以大致分为算术操作符、逻辑操作符、比较操作符等几大类，见表 9.2.1。

表 9.2.1 Verilog HDL 的操作符和简要说明

分类	操作符及功能	简要说明
算术操作符	+ 加 - 减 * 乘 / 除 % 整除	二元操作符，即有两个操作数。操作数可以是物理数据类型，也可以是抽象数据类型。
比较操作符	> 大于 < 小于 >= 不小于 <= 不大于 == 相等 != 不相等 === 全等 !== 非全等	二元操作符，如果操作数之间的关系成立，返回值为 1；关系不成立，则返回值为 0。若某一个操作数的值不定，则关系是模糊的，返回值是不定值 X。

续表

分类	操作符及功能		简要说明
逻辑操作符	&&	逻辑与	&& 和 ‖ 为二元操作符；! 为一元操作符，即只有一个操作数。
	‖	逻辑或	
	!	逻辑非	
位操作符	~	按位非	“~”是一元操作符，其余都是二元操作符。将操作数按位进行逻辑运算。
	&	按位与	
	|	按位或	
	^	按位异或	
	^~(~^)	按位同或	
归约操作符	&	归约与	一元操作符，对操作数各位的值进行运算。如“&”是对操作数各位的值进行逻辑与运算，得到一个一位的结果值。
	~&	归约与非	
	|	归约或	
	~|	归约或非	
	^	归约异或	
	~^(或^~)	归约同或	
移位操作符	>>	左移	二元操作符，对左侧的操作数进行它右侧操作数指明的位数的移位，空出的位用 0 补全。
	<<	右移	
条件操作符	?:		三元操作符，即条件操作符有三个操作数。 如 a? b:c 若第 1 个操作数 a 是逻辑 1，则算子返回第 2 个操作数 b；若 a 是逻辑 0，则算子返回第 3 个操作数 c。
连接和复制符	{,}		将两个或两个以上用逗号分隔的表达式按位连接在一起。 还可以用常数来指定重复的次数，如{a,{2{a,b}}}等价于{a,a,b,a,b}。

同其他高级语言类似，各类操作符号之间有优先级之分，如下所示。列表顶部是最高优先级，底部是最低优先级。列在同一行中的操作符具有相同的优先级。所有操作符(?:除外)在表达式中都是从左向右结合的。圆括号( )用于改变优先级或使得表达式中运算顺序更加清晰，提高源文件的可读性。

```
!    &    ~&    |    ~|    ^    ~^    +    -          最高优先级
*    /    %
>>    <<
<    >    >=    <=
==    !=    ===    !==
&    ~&    ^    ~^
|    ~|
&&
||
?:                                                    最低优先级
```

三、数值常量

Verilog HDL 中的数值常量有整型和实型两大类，分为十进制、十六进制、八进制或二进制。若在前面加上一个正“+”或负“-”号就表示有符号数，否则所代表的就是无符号数。在数值常量的任意位置可以随意插入下划线“_”以提高可读性。

Verilog HDL 中的整型数值常量就是整数，有两种书写格式：一种是无位宽的十进制表示法，如 -132。第二种是定义位宽和进制的表示法，这种表示方法通常是无符号数，书写格式是：[size]'base value。其中 size 是可选项，定义了数值常量的位数（长度）；base 代表这个数据的进制，取值范围和相应的进制见表 9.2.2；value 是一个数值常量的值，书写格式与进制 base 相对应。如 4'h6a8c，它表示一个 4 位的十六进制数。

表 9.2.2 Verilog HDL 中的进制

base 进制代码取值	对应的进制
b 或 B	二进制
o 或 O	八进制
d 或 D	十进制
h 或 H	十六进制

Verilog HDL 中的实型数值常量就是浮点数，可以用十进制与科学计数法两种形式书写。如果采用十进制格式，小数点两边必须都有数字。

Verilog HDL 的编程最终是与硬件相对应的。由于硬件电路中信号的逻辑状态具有特殊性，即不仅有 0（低电平）和 1（高电平），还有可能是 X（未知状态）

和 Z(高阻态),因此 Verilog HDL 数值集合有四个基本值:

0:逻辑 **0** 或假状态;

1:逻辑 **1** 或真状态;

X:逻辑不定态;

Z:高阻态。

四、字符串

字符串是双引号“”括起来的字符序列,必须包含在一行中,不能多行书写。在表达式或赋值语句中作为操作数的字符串被看作 ASCII 值序列,即一个字符串中的每一个字符对应一个 8 位 ASCII 值。

五、标识符

标识符是模块、寄存器、端口、连线、示例和 begin - end 块等元素的名称,是赋给对象的唯一的名称。标识符可以是字母、数字、$ 符和下划线“_”字符的任意组合序列,必须以字母或下划线“_”开头。在 Verilog HDL 中,标识符区分大小写,且字符数不能多于 1024。

六、关键词

关键词是 Verilog HDL 语言内部的专用词。在 IEEE 标准——Verilog HDL 1364 - 1995 中规定了 102 个关键词,都采用小写形式。作为语言本身的保留字,关键词有其特定和专有的语法作用,用户不能再对它们做新的定义,见表9.2.3。

表 9.2.3 关 键 词

always	else	highzl	notif1	rtranif1	tri
and	end	if	or	scalared	tri0
assign	endattribute	initia1	output	signed	tri1
attribute	endcase	inout	parameter	small	triand
begin	endfunction	input	pmos	specify	trior
buf	endmodule	integer	posedge	specpram	triteg
bufif0	endprimitive	join	primitive	strength	unsigned
bufif1	endspecify	large	pull0	strong0	vectored
case	endtable	macromodule	pull1	strong1	wait
casex	endtask	medium	pulldown	supply0	wand
casez	event	module	pullup	supply1	weak0
cmos	for	nand	rcmos	table	weak1
deassign	force	negedge	real	task	while

续表

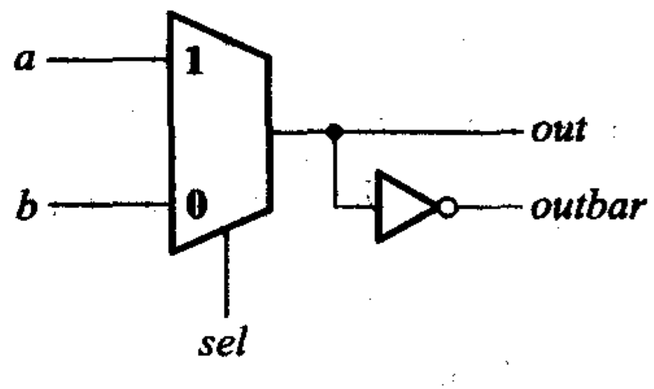
default	forever	nmos	realtime	time	wire
defparam	fork	nor	reg	tran	wor
disable	function	not	release	tranif0	xnor
edge	highz0	notif0	repeat	tranif1	xor

9.2.3 模块的两种描述方式

一、行为描述方式

行为描述方式和其他软件编程语言的描述方式类似,通过行为语句来描述电路要实现的功能,表示输入与输出间转换的行为,不涉及具体结构。从这个意义上讲,行为建模是一种"高级"的描述方式。

以图9.2.1所示的2选1数据选择器为例,若用 Verilog HDL 对它做行为描述,则可写成下面的程序模块。



$Out=sel\cdot a+sel'\cdot b$

图9.2.1 2选1数据选择器

```
module  mux_2_to_1(a,b,out,outbar,
sel);
                                    //这是一个2选1数据选择器,名为 mux_2_to_1
  input   a,b,sel;                  //定义该模块的输入端口为 a,b 和 sel
  output  out,outbar;               //定义该模块的输出端口为 out 和 outbar
  assign  out = sel? a:b;           //如果 sel = 1,将 a 赋值给 out
                                    //如果 sel = 0,将 b 赋值给 out
  assign  outbar = ~out;            //将 out 取反后赋值给 outbar
endmodule                           //模块描述结束
```

二、结构描述方式

结构描述方式是将硬件电路描述成一个分级子模块相互连的结构。通过对组成电路的各个子模块间相互连接关系的描述,来说明电路的组成。各个模块还可以对其他模块进行调用,也就是模块的实例化。其中调用模块成为层次结构中的上级模块,被调用模块成为下级模块。从结构上而言,任何硬件电路都是由一级级的不同层次的若干单元组成,因此结构描述方式很适合对电路的这种层次化结构的描述。在结构描述中,门和 MOS 开关是电路最低层的结构。在 Verilog HDL 中有26个内置的基本单元,又称基元,见表9.2.4.

表 9.2.4　Verilog HDL 中的基元

基元分类	基元
多输入门	and, nand, or, nor, xor, xnor
多输出门	buf, not
三态门	bufif0, bufif1, notif0, notif1
上拉、下拉电阻	pullup, pulldown
MOS 开关	cmos, nmos, pmos, rcmos, rnmos, rpmos
双向开关	tran, tranif0, tranif1, rtran, rtranif0, rtranif1

仍以 2 选 1 数据选择器为例，若给出它的门级电路原理图如图 9.2.2 所示，采用结构描述方式可以写成下面的程序模块。

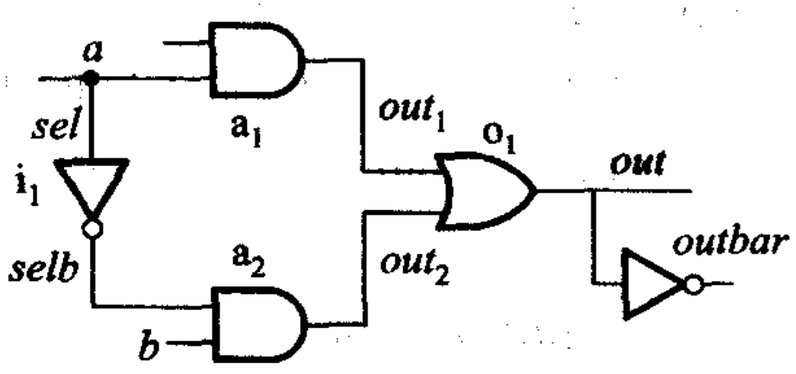


图 9.2.2　2 选 1 数据选择器的电路原理图

```
module muxgate(a,b,out,outbar,sel);
//这是一个 2 选 1 数据选择器,名为 muxgate
    input   a,b,sel;    //定义输入端口为 a,b 和 sel
    output  out,outbar;  //定义输出端口为 out 和 outbar
    wire    out1,out2,selb;  //定义内部的两个连接点 out1,out2,selb
        and   a1(out1,a,sel);   //调用一个与门 a1
        not   i1(selb,sel);     //调用一个反相器 i1
        and   a2(out2,b,selb);  //调用一个与门 a2
        or    o1(out,out1,out2);  //调用一个或门 o1
        assign  outbar = ~out;
endmodule
```

9.3　用 Verilog HDL 描述逻辑电路的实例

在这一节里，我们再通过两个简单的例子来说明用 Verilog HDL 描述电路逻辑功能的方法。

【例 9.3.1】　用 Verilog HDL 对图 9.3.1 所示的 4 位加法器做逻辑功能描述。

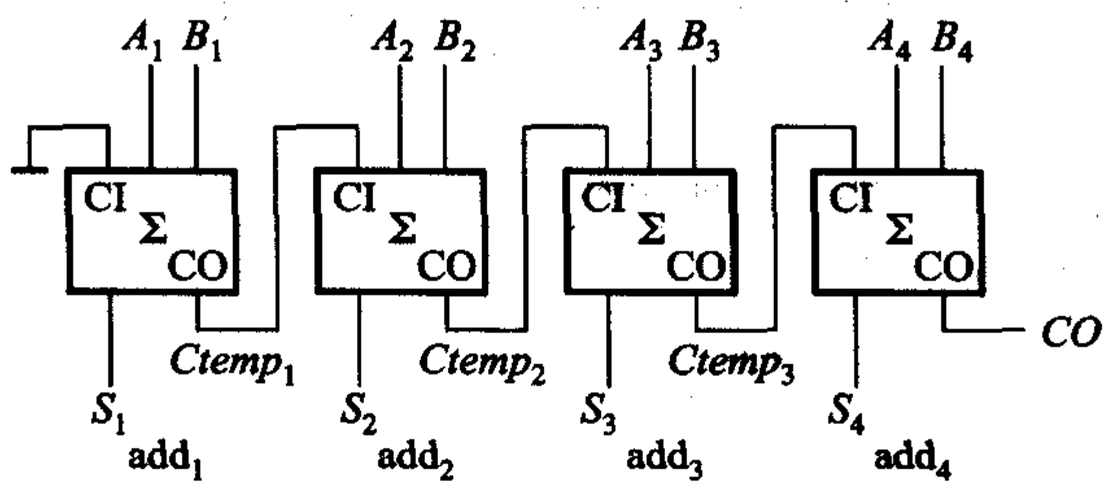


图 9.3.1 例 9.3.1 的 4 位加法器

加法器是常用的组合电路之一,本例描述了第 4 章中用串行进位方法构成的 4 位全加器,如图 4.3.28 所示。

```
//对 4 位串行进位加法器的顶层结构的描述
module  Four _ bit _ fulladd(A,B,CI,S,CO);
                                    //4 位全加器模块名称和端口名
    parameter  size =4;                          //定义参数
    input  [size:1]A,B;
    output  [size:1]S;
     input  CI;
     output  CO;
     wire  [1:size -1]Ctemp                      //定义模块内部的连接线
onebit _ fulladd                                 //调用 1 位全加器
    add1(A[1],B[1],CI,  S[1],Ctemp[1]),
                                    //实例化,调用 1 位全加器
    add2(A[2],B[2],Ctemp[1],  S[2],Ctemp[2]),
                                    //实例化,调用 1 位全加器
    add3(A[3],B[3],Ctemp[2],  S[3],Ctemp[3],)
                                    //实例化,调用 1 位全加器
    add2(A[4],B[4],Ctemp[3],  S[4],CO);  //实例化 4
endmodule  //结束
```

上面的程序仅对图 9.3.1 所示电路进行了顶层描述,在程序中调用 1 位全加器 onebit _ fulladd。如采用 1 位全加器,用图 9.3.2 所示的方式实现,则可以用下面的程序模块进行描述。

```
//对 1 位全加器内部结构的描述
module  onebit _ fulladd(A,B,CI,Sum,Cout);
                                    //1 位全加器模块名称和端口名
```

```
        input   A,B,CI;
        output  Sum,Cout;
        wire    Sum_temp,C_1,C_2,C_3;
//定义模块内部的连接线
        xor
        XOR1(Sum_temp,A,B),
        XOR2(Sum,Sum_temp,CI);
//两次调用异或门实现 Sum = A⊕B⊕CI
        and   //调用3个与门 AND1,AND2,AND3
        AND1(C_1,A,B),
        AND2(C_2,A,CI),
        AND3(C_3,B,CI);
        or
        OR1(Cout,C_1,C_2,C_3);
                //调用或门实现 Cout = AB + A(CI) + B(CI)
endmodule   //结束
```

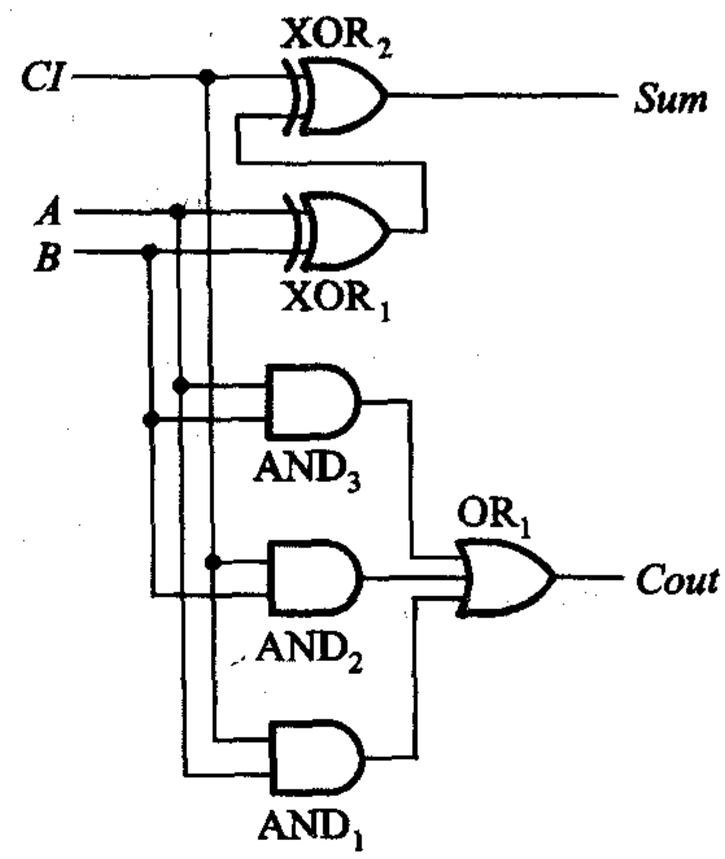


图 9.3.2 例 9.3.1 中的 1 位全加器电路

【例 9.3.2】 用 Verilog HDL 描述图 9.3.3 所示状态转换图实现的逻辑功能。

第 6 章所介绍的状态转换图是时序电路的一种通用模型,在 Verilog HDL 中可以直接对状态转换图所描述的有限状态机进行描述,从而实现对电路的逻辑功能的描述。下面的程序模块就是用 Verilog HDL 对图 9.3.3 所示状态转换图的逻辑功能的描述。

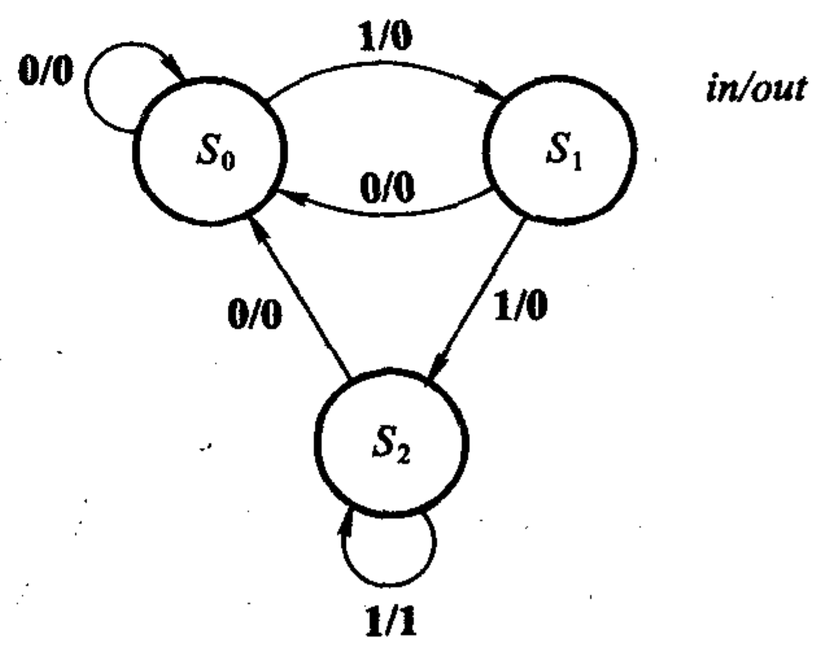


图 9.3.3 例 9.3.2 的状态转换图

```
    module  fsm(out,in,clock);   //模块名称 fsm,模块端口列表 out,in,clock
        output  out;   //定义输出
        input   in,clock;   //定义输入
        reg   out;   //定义输出变量的类型
        reg   [1:0]currentState,nextState;
```