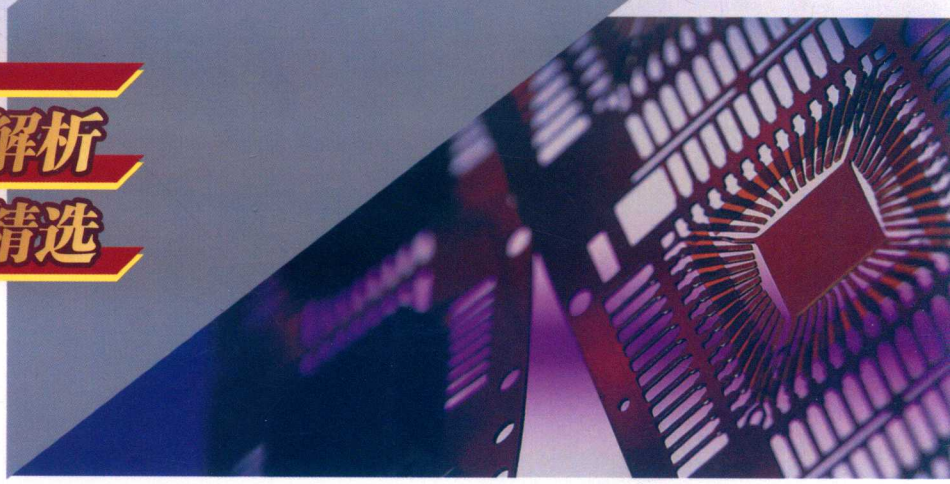
大学课程学习与考研
全程辅导系列丛书

考研宝典

数字电子技术基础
考研大串讲

主　编　杨春玲　陶隽源
副主编　张　岩

>> 名师辅导
>> 典型例题解析
>> 考研试题精选



科学出版社

需要全本请访问 www.ertongbook.com

大学课程学习与考研全程辅导系列丛书

数字电子技术基础考研大串讲

主　编　杨春玲　陶隽源

副主编　张　岩

安徽理工大学 藏書 图書馆

科学出版社

北　京

内 容 简 介

本书是高等院校电气信息、电子信息科学类专业“数字电子技术基础”课程的考研指导书。书中对“数字电子技术基础”课程的内容进行概括和总结，给出需要掌握的要点及考研考点，并且精解了部分重点高校近年来研究生入学考试试题中对应考点的例题，每章后面给出了一些习题及详细解答，最后给出部分重点高校研究生入学考试试卷。

本书可以作为“数字电子技术基础”课程的教学辅导书，也可以作为工程技术人员的参考书。

图书在版编目(CIP)数据

数字电子技术基础考研大串讲 / 杨春玲，陶隽源主编. —北京：科学出版社，2019.12

大学课程学习与考研全程辅导系列丛书

ISBN 978-7-03-061508-4

Ⅰ. ①数… Ⅱ. ①杨… ②陶… Ⅲ. ①数字电路－电子技术－研究生－入学考试－自学参考资料 Ⅳ. ①TN79

中国版本图书馆 CIP 数据核字(2019)第 109879 号

责任编辑：潘斯斯 张丽花 霍明亮 / 责任校对：郭瑞芝

责任印制：张 伟 / 封面设计：迷底书装

科 学 出 版 社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

石家庄继文印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2019 年 12 月第 一 版 开本：787×1092 1/16

2019 年 12 月第一次印刷 印张：14

字数：332 000

定价：49.00 元

(如有印装质量问题，我社负责调换)

前　言

本书是为了配合“数字电子技术基础”课程学习编写的辅导书，既可以作为研究生入学考试的复习参考资料，也可以作为学习数字电子技术基础课程的辅助教材。全书由重点内容、考研考点、例题精解、习题及解答以及研究生入学考试试卷等部分组成。

本书逐章讲解“数字电子技术基础”课程的重点及学习要求，对学生在学习中容易产生疑问的主要难点进行论述，给出需掌握的要点；给出了每章的考研考点，并对以往一些典型的考研试题给出详细的解答；给出了一些习题及详细解答；最后给出部分重点高校研究生入学考试试卷。由于设计性题目的答案往往不是唯一的，所以书中给出的这类题目的答案可能只是其中的一种，不能以此作为判断正、误的唯一标准。

参加本书编写工作的有杨春玲、陶隽源、张岩、王淑娟、刘贵栋、齐明、吕超、王立欣、于泳、杨荣峰、朱敏、徐乐和张刚。全书由杨春玲和陶隽源任主编，张岩任副主编。

编者衷心期望本书提供的内容能够对学生在更好地掌握课程内容、提高解题能力以及系统复习迎接考试等方面有所帮助。本书编写力求条理清晰，语言准确，文字简洁，图表规范。

由于水平有限，书中不妥之处在所难免，恳请读者给予批评指正。

编　者

于哈尔滨工业大学

2019 年 6 月

目　录

第 1 章　Verilog 语言 …… 1

1.1　名师辅导 …… 1
1.2　例题精解 …… 16
1.3　习题及解答 …… 23

第 2 章　数制和码制 …… 32

2.1　名师辅导 …… 32
2.2　例题精解 …… 33
2.3　习题及解答 …… 34

第 3 章　逻辑代数基础 …… 35

3.1　名师辅导 …… 35
3.2　例题精解 …… 39
3.3　习题及解答 …… 42

第 4 章　门电路 …… 47

4.1　名师辅导 …… 47
4.2　例题精解 …… 54
4.3　习题及解答 …… 60

第 5 章　组合逻辑电路 …… 66

5.1　名师辅导 …… 66
5.2　例题精解 …… 74
5.3　习题及解答 …… 83

第 6 章　触发器 …… 93

6.1　名师辅导 …… 93
6.2　例题精解 …… 98
6.3　习题及答案 …… 102

第 7 章　时序逻辑电路 …… 108

7.1　名师辅导 …… 108
7.2　例题精解 …… 120
7.3　习题及答案 …… 137

需要全本请访问 www.ertongbook.com

第 8 章　半导体存储器 …… 147
8.1　名师辅导 …… 147
8.2　例题精解 …… 153
8.3　习题及解答 …… 154
第 9 章　脉冲产生及变换电路 …… 157
9.1　名师辅导 …… 157
9.2　例题精解 …… 163
9.3　习题及解答 …… 169
第 10 章　数模与模数转换器 …… 174
10.1　名师辅导 …… 174
10.2　例题精解 …… 179
10.3　习题及解答 …… 180
第 11 章　试卷及其参考答案 …… 185
11.1　试卷 1 及其参考答案 …… 185
11.2　试卷 2 及其参考答案 …… 190
11.3　试卷 3 及其参考答案 …… 195
11.4　试卷 4 及其参考答案 …… 202
11.5　试卷 5 及其参考答案 …… 207
参考文献 …… 213
附录　涉及数字集成电路的功能表 …… 214

第1章 Verilog语言

1.1 名师辅导

本章重点掌握 Verilog 语言的语法规定及使用 Verilog 语言描述组合和时序逻辑电路。Verilog 语言描述逻辑电路时常用 3 种描述方式，分别为行为型描述、数据流型描述和结构型描述，一个实例可以是这三种描述的组合。通过本章的学习还能掌握典型状态机的 Verilog 描述方法以及连续赋值、阻塞赋值和非阻塞赋值的区别。

1.1.1 重点内容

1. Verilog 语言描述组合逻辑电路

1)2 选 1 数据选择器实例 1

Verilog 语言描述：

```
//例 1.1.1
module Mux21 (a,b,s,y);                    //----------------1
   input a,b;
   input s;
   output y;
   assign y=(s==0)? a:b;                   //----------------2
endmodule                                  //----------------3
```

语法规定：

(1) Verilog 语言程序结构。

```
module <模块名> (端口列表)
   <申明>
   <功能描述>
Endmodule
```

程序行 1 和程序行 3 是模块定义的开始和结尾。模块定义语句 module 和 endmodule 所有的程序都置于模块(module)框架结构内。

模块名是该模块的唯一标识符。端口列表列举了该模块与外部电路连接的所有端口，包括输入、输出及双向端口。

申明部分又包括 I/O 端口申明和模块内部用到的变量的申明；功能描述部分定义了模块的功能，是模块最重要的部分。

Verilog 端口类型只有 input(输入)、output(输出)和 inout(双向端口)三种，端口类型申明描述了端口的信号传输方向。图 1.1.1 为 Verilog 的端口类型。

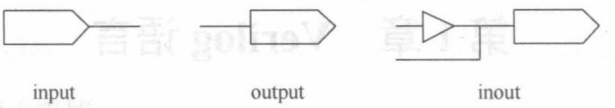


图 1.1.1　Verilog 的端口类型

(2)注释行。Verilog 中可以用“//”进行单行注释。注释行会被编译器和仿真器所忽略。

(3)连续赋值语句 assign。连续赋值语句 assign 能够给网表变量赋值。只要等号右边的表达式值发生变化，这种赋值行为就会立刻发生。连续赋值语句能模拟组合逻辑电路。

(4)条件操作符“?”。程序行 2 中(s==0)?a:b 为一个条件表达式。式中条件操作符“?”为三目操作符，由两个操作符隔离三个操作数构成，语法格式如下：

```
表达式 1? 表达式 2: 表达式 3
```

执行操作时，首先会计算表达式 1 的值，如果表达式 1 的值为 1，那么将计算表达式 2 的值，作为条件表达式的最后结果；如果表达式 1 的值为 0，则计算表达式 3 的值，并作为条件表达式最后的结果。

(5)等式运算符“==”。程序行 2 中表达式 s==0 中的“==”为等式运算符，当“==”两边相等时，结果为 1，否则为 0。“!=”为不等式运算符。

因此程序的功能为当 s 为 0 时，y 状态和 a 相同，否则 y 状态和 b 相同，实现了 2 选 1 数据选择器功能。

2)2 选 1 数据选择器实例 2

Verilog 语言描述：

```
//例 1.1.2
module Mux21 (a,b,s,y);
   input a,b;
   input s;
   output y;
   wire d,e;
   assign d=a&(~s);
   assign e=b&s;
   assign y=d|e;
endmodule
```

语法规定：

(1)连线类型 wire。wire 类型表示内部节点或连线。网表变量可通过连续赋值语句(assign 语句)或逻辑门驱动，如果一个网表变量没有和任何驱动源相连，则其值为高阻态(z)。

(2)位运算符。连续赋值语句 assign 里出现了位运算符取反(~)、按位与(&)和按位或(|)。Verilog 中位运算符是对操作数的每一位进行操作的运算符，共包括 5 个位运算符，除上述 3 个，还包括按位异或(^)、按位同或(^~，异或非)。位运算规则如表 1.1.1 所示。

表 1.1.1　位运算规则

&	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

|	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

^	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

~	—
0	1
1	0
x	x
z	x

^~	0	1	x	z
0	1	0	x	x
1	0	1	x	x
x	x	x	x	x
z	x	x	x	x

两个长度不同的数据进行位运算时，系统会自动地将两者按右端对齐。位数少的操作数会在相应的高位用 0 填满，以使两个操作数按位进行操作。

3)2 选 1 数据选择器实例 3

Verilog 语言描述：

```
//例 1.1.3
module mux21 (a,b,s,y);
   input a,b,s;
   output y;
   reg y;
      always @( a or b or s)   //------------------------------1
            if (!s)  y=a;      //------------------------------2
            else y=b;          //------------------------------3
endmodule
```

语法规定：

(1)寄存器类型 reg。在例 1.1.3 中，输出端口 y 的数据类型申明为 reg 即寄存器型，在 always 块中被赋值的信号必须定义为 reg 型。

(2)重复执行语句 always。程序行 1 中 always 后面跟了一个时间控制语句，时间控制通过事件表达式(关键词“@”)实现，时间控制部分为完整敏感信号列表，只要任意敏感信号发生变化，过程块将重复连续执行，持续整个模拟过程。always 语句具有循环特性。

程序中应尽量采用完整电平敏感列表方式，否则综合出来的电路功能可能与程序模块的描述有所不同。

(3)条件语句 if-else。条件语句 if-else 可用来选择是否执行某条语句。其语法结构为

```
if (表达式)
   语句 1;
else
   语句 2;
```

如果表达式值为 1，则执行语句 1，否则执行语句 2。因此，程序行 2 和 3 的作用是，如果 s 等于 0，y 和 a 相同，如果 s 等于 1，则 y 和 b 相同。if-else 只能出现在 always 语句中。

(4)逻辑运算符。在 Verilog HDL 语言中存在三种逻辑运算符：逻辑与“&&”、逻辑或“||”和逻辑非“!”。逻辑表达式的值可为 1(真)和 0(假)。

“&&”和“||”是二目运算符，它要求有两个操作数，如(a>b)&&(b>c)、(a<b)||(b<c)。“!”是单目运算符，只要求有一个操作数，如!(a>b)。表 1.1.2 为逻辑运算的真值表。它给出当 a 和 b 的值为不同组合时，各种逻辑运算所得到的值。

表 1.1.2　逻辑运算的真值表

a	b	!a	!b	a&&b	a||b
真	真	假	假	真	真
真	假	假	真	假	真
假	真	真	假	假	真
假	假	真	真	假	假

逻辑运算符中“!”的优先级别高于“&&”和“||”，逻辑运算符的优先级别低于关系运算符，括号优先级最高，例如：(a>b)&&(x>y)可写成 a>b && x>y；(a==b)||(x==y)可写成 a==b || x==y；(!a)||(a>b)可写成!a || a>b。

4)2 选 1 数据选择器实例 4

Verilog 语言描述：

```
//例 1.1.4
module mux21(y, a, b, s);
   input a, b,s;
   output y;
   not u1 (ns,s);
   and u2 (sela,a,ns);
   and u3 (selb,b,s);
   or u4 (y,sela,selb);
endmodule
```

语法规定：

例 1.1.4 调用了多个底层模块的实例。调用模块的过程，称为实例化。实例的使用格式为

```
<模块名> <实例名> <端口列表>;
```

例 1.1.4 调用了非门、与门和或门的实例。

Verilog HDL 中一些基元与关键字对应关系如下：

not——非门；

buf——缓冲器；
and——与门；
or——非门；
nand——与非门；
nor——或非门；
xor——异或门；
xnor——同或门。

5)4 选 1 数据选择器实例 1

Verilog 语言描述：

```
//例 1.1.5
module mux41(out,in0,in1,in2,in3,sel);
   output out;
   input in0,in1,in2,in3;
   input[1:0] sel;
   reg out;
   always @(in0 or in1 or in2 or in3 or sel)
   case(sel)
            2'b00: out=in0;
            2'b01: out=in1;
            2'b10: out=in2;
            2'b11: out=in3;
            default: out=1'bx;
   endcase
endmodule
```

语法规定：

(1)矢量类型。“input [1:0] sel;”语句定义了一个矢量网表。包括 sel[1]和 sel[0]两个变量。Verilog 中如果一个网表型变量和寄存器型变量定义时没有指定位长度，则它被认为是 1 位标量，如果设定了位长度，则被认为是一个矢量。例如：

```
wire [7:0] bus;              //8 位矢量网表 bus
reg [0:40] addr;             //41 位矢量寄存器 addr
```

Verilog 中矢量被当作无符号数处理。矢量的定义可以是[高位:低位]，也可以是[低位:高位]。赋值语句中，如果赋值对象为一个标量类型的变量，而欲赋的值为一个矢量，则将矢量的最低有效位赋值给该变量。对矢量进行操作，相当于对矢量包括的每个变量进行操作。

(2)数的表示方法。按进制划分，整数可以表示成十进制数、十六进制数、八进制和二进制数。Verilog 中整数通常有两种表述方式，分别为十进制数和指定位数表述方式。

十进制数表述方式：用 0～9 的数字序列表示。

指定位数表述方式：<位长度><'进制符号><数字及 a～f(十六进制)>。

指定位数表述方式中位长度可缺省，若不指定位长度，则系统默认位长度为 32。位长度定义了数据的确切位(bit)数，应为无符号的十进制数。

进制符号可以是 b 或 B(二进制)，d 或 D(十进制)，h 或 H(十六进制)，o 或 O(八进制)。

需要全本请访问 www.ertongbook.com

采用这种表示方法，还必须在进制符号前加“ ' ”号，并且“ ' ”号和进制符号间不能存在空格。数字为无符号数，应该与进制格式一致，数字与进制符号之间可以有空格。

在硬件描述语言中，x 表示不定值，z 表示高阻态。可以在十六进制、八进制和二进制中使用 x 和 z，十六进制中一个 x 表示 4 位都是 x，八进制中一个 x 表示 3 位都是 x，二进制中则表示一位是 x。z 用法同理。

在表示长数据时还可以用下划线“_”进行分割以增加程序的可读性。表示数据时“_”将被忽略，但不能放在数据的开头。

(3) case 语句的语法结构。case 语句是分支条件语句，case 语句语法结构为

```
case (表达式)
   选项值 1: 语句 1;
   选项值 2: 语句 2;
   选项值 3: 语句 3;
   …
default: 缺省语句;
endcase
```

执行 case 语句时，先计算表达式的值，按照各选择项出现的先后顺序，比较不同选择项的值和 case 表达式的值，找到首先匹配的项，执行对应语句。如果没有选项的值和表达式的值匹配，则执行缺省语句。缺省语句是可选而不是必需的，并且不能有多条缺省语句，如果没有缺省语句并且没有选项匹配，则不会有 case 项语句执行。

例 1.1.5 程序的功能为当 sel 为 0、1、2、3 时，输出 out 分别等于 in0、in1、in2、in3，实现了 4 选 1 数据选择器的功能。

6) 4 选 1 数据选择器实例 2

Verilog 语言描述：

```
//例 1.1.6
module mux41(out,in0,in1,in2,in3,sel);
   output out;
   input in0,in1,in2,in3;
   input[1:0] sel;
   reg out;
   always @(in0 or in1 or in2 or in3 or sel)
   begin
       if(sel==2'b00) out=in0;
       else if(sel==2'b01) out=in1;
       else if(sel==2'b10) out=in2;
       else if(sel==2'b11) out=in3;
   else
       out=1'bx;
   end
endmodule
```

语法规定：

(1) if 条件语句除了 if-else 结构，还有 if-else-if 结构，并且应用很广，其语法结构为

```
if (表达式 1)
   语句 1;
else if (表达式 2)
   语句 2;
else if …
else
   语句 n;
```

if-else-if 结构将顺序计算并判断表达式的值，如果表达式值为真，则执行相应的语句，并且终止整个条件语句。

(2) 顺序块语句 begin-end。顺序块语句通常用来将两条或多条语句组合在一起，使其在格式上更像一条语句。

顺序块有以下特点：①块内的语句是按顺序执行的，即只有上面一条语句执行完后下面的语句才能执行。②直到最后一条语句执行完，程序才跳出该语句块。

(3) 缺省项问题。条件语句和分支条件语句都存在着缺省项的问题。对于条件语句，else 语句称为缺省项，对于 case 语句，default 语句称为缺省项。缺省项是可以省略的，但省略缺省项会引入锁存器，在组合逻辑电路设计中可能会带来一些问题。

if 条件语句和 case 条件语句在省略缺省项时，会产生锁存器。因此，在设计组合逻辑电路时缺省项不能省略。

7) 4 位加法器

Verilog 语言描述：

```
//例 1.1.7
//全加器
module add_full(A,B,C,Carry,S);
   input A,B,C;
   output Carry,S;                          //全加器进位及和
   assign S=A^B^C;
   assign Carry=(A&B)|(B&C)|(A&C);
endmodule
//四位加法器
module add_full4(A,B,C,S);
   input[3:0] A,B;
   output[3:0] S;                           //加法器和
   output[4:0] C;                           //加法器进位
   assign  C[0]=0;
   add_full u1(A[0],B[0],C[0],C[1],S[0]), //-----------------------1
            u2(A[1],B[1],C[1],C[2],S[1]),
            u3(A[2],B[2],C[2],C[3],S[2]),
            u4(A[3],B[3],C[3],C[4],S[3]);
endmodule
```

语法规定：

(1)实例。4位加法器的功能是实现2个4位二进制数的加法运算。4位加法器由两个模块组成，分别为底层全加器模块和顶层模块，顶层模块处于整个系统的最上层，在其内部可以调用多个底层模块实例。

(2)实例的调用。顶层模块add_full4调用了底层模块实例add_full的过程称为实例化。

实例的调用格式1：

```
<模块名> <实例名> <端口列表>;
```

add_full u1(A[0],B[0],C[0],C[1],S[0])表明调用了一个add_full的底层模块实例，实例名为“u1”，对应接口有A[0]、B[0]、C[0]、C[1]和S[0]。一条语句可以多次调用某个模块的实例，例1.1.7中调用了4次全加器模块，采用这种实例化方式，应注意保持端口列表里端口的顺序和模块申明时端口的顺序一致。

实例的调用格式2：

```
<模块名> <实例名> <.实例端口1(模块端口1),.实例端口2(模块端口2)>;
```

实例调用格式2是通过实例端口和模块端口直接映射的方式进行引用的，例如：

```
add_full u1(.A(A[0]),.B(B[0]),.C(C[0]),.Carry(C[1]),.S(S[0])),
         u2(.A(A[1]),.B(B[1]),.C(C[1]),.Carry(C[2]),.S(S[1])),
         u3(.A(A[2]),.B(B[2]),.C(C[2]),.Carry(C[3]),.S(S[2])),
         u4(.A(A[3]),.B(B[3]),.C(C[3]),.Carry(C[4]),.S(S[3]));
```

这种实例引用格式，端口出现次序可以不用考虑实例申明时的次序，避免出错。而且采用这种方式时，如果有的端口并没有和外部连接，则可以直接省略该端口。

注意模块的定义与实例的关系：Verilog中不允许嵌套定义模块，即一对module和endmodule之间只能定义一个模块。但一个模块内可以通过实例的方式多次调用其他模块。4位加法器由4个全加器构成，实现了4位二进制加法运算。

8)七段数码管显示译码器

Verilog语言描述：

```
//例1.1.8
module decode48(a,b,c,d,e,f,g,D3,D2,D1,D0);
   output a,b,c,d,e,f,g;
   input D3,D2,D1,D0;                          //输入4位BCD码
   reg a,b,c,d,e,f,g;                          //输出驱动共阳极7个笔画段
   always @(D3 or D2 or D1 or D0)
   begin
      case({D3,D2,D1,D0})                      //用case语句进行译码
         4'd0: {a,b,c,d,e,f,g}=7'b1111110;
         4'd1: {a,b,c,d,e,f,g}=7'b0110000;
         4'd2: {a,b,c,d,e,f,g}=7'b1101101;
         4'd3: {a,b,c,d,e,f,g}=7'b1111001;
         4'd4: {a,b,c,d,e,f,g}=7'b0110011;
         4'd5: {a,b,c,d,e,f,g}=7'b1011011;
```

```
            4'd6: {a,b,c,d,e,f,g}=7'b1011111;
            4'd7: {a,b,c,d,e,f,g}=7'b1110000;
            4'd8: {a,b,c,d,e,f,g}=7'b1111111;
            4'd9: {a,b,c,d,e,f,g}=7'b1111011;
            default: {a,b,c,d,e,f,g}=7'bx;
        endcase
    end
endmodule
```

语法规定：

拼接操作符({　})：拼接操作符将两个或者更多的表达式结果的各个位进行连接。其表达式由大括号中的用逗号分隔的表达式构成。

```
{表达式 1,表达式 2,…}
```

例如：

```
{a,b[1:2],c ,4'b0010}
{4{y}}
```

第一行等同于{a,b[1],b[2],c,1'b0,1'b0,1'b1,1'b0}，第二行是嵌套反复多次调用，它等同于{y,y,y,y}。

本实例逻辑功能为当输入 D3、D2、D1、D0 为 0001 时，输出 a、b、c、d、e、f 为 011000，数码管显示 1，例 1.1.8 可以实现显示译码器的功能。

2. Verilog 语言描述时序逻辑电路

1)D 触发器

Verilog 语言描述：

```
//例 1.1.9
module d_ff (q,d,clk);
   output q;
   input d,clk;
   reg q;
   always @(posedge clk)
      q=d;
endmodule
```

语法规定：

(1)时钟上升沿检测语句。上升沿触发指变量值从 0 变为 1、0 变为 x 和 z 或者从 x、z 变为 1，用 posedge 表示。

(2)时钟下降沿检测语句。下降沿触发指变量值从 1 变为 0、1 变为 x 和 z 或者从 x、z 变为 0，用 negedge 表示。上升沿和下降沿检测语句的功能表如表 1.1.3 所示。

表 1.1.3　上升沿和下降沿检测语句的功能表

From/To	0	1	x	z
0	No	posedge	posedge	posedge
1	negedge	No	negedge	negedge
x	negedge	posedge	No	No
z	negedge	posedge	No	No

(3) Verilog 语言的赋值方式。Verilog 语言有两种赋值方式：连续赋值和过程赋值。

连续赋值需要将关键词 assign 用于驱动网表变量，一旦输入操作数改变，数据便会更新。

过程赋值用来更新寄存器类型变量的值，也就是说对出现在 always 语句块中的变量进行赋值，包括阻塞赋值“=”和非阻塞赋值“<=”。

阻塞赋值“=”：这种赋值方式是立即执行。前面的赋值语句阻塞后面的赋值语句，也就是说语句是先后执行的。

综上所述，过程赋值包括阻塞赋值和非阻塞赋值两种，当出现在 always 语句块内，阻塞赋值是顺序执行，非阻塞是同时执行。

2) 4 位二进制加法计数器

Verilog 语言描述：

```
//例 1.1.10
module count4(out,reset,clk);
   output[3:0] out;
   input reset,clk;
   reg[3:0] out;
   always @(posedge clk)
   begin
       if (reset) out<=0;                    //同步清零
       else out<=out+1;                      //计数
   end
endmodule
```

程序说明：

这个计数器程序比较短小，功能上也比较简单，只有同步复位和计数功能。时钟的上升沿有效，当 clk 信号的上升沿到来时，如果清零信号为 1，则计数器清零，否则计数器进行计数。例 1.1.10 实现了 4 位二进制加法计数器的逻辑功能。当 reset 为高电平时，计数器清零。

3) 同步置数同步清零加法计数器

Verilog 语言描述：

```
//例 1.1.11
module count(out,data,load,reset,clk);
   output[7:0] out;
   input[7:0] data;
   input load,clk,reset;
   reg[7:0] out;
   always @(posedge clk)                          //clk 上升沿触发
```

```
    begin
        if (!reset) out <=8'h00;                    //同步清零，低电平有效
            else if (!load) out <=data;             //同步预置
            else out <=out + 1;                     //计数
    end
endmodule
```

程序说明：

例 1.1.11 为同步置数、同步清零加法计数器。表 1.1.4 为例 1.1.11 计数器功能表。

表 1.1.4　例 1.1.11 计数器功能表

clk	reset	load	out
↓	0	x	清零
↓	1	0	置数(data)
↓	1	1	计数

4) 异步清零计数器

Verilog 语言描述：

```
//例 1.1.12
module count2(out,reset,clk);
    output[7:0] out;
    input clk,reset;
    reg[7:0] out;
    always @(posedge clk or negedge reset)
    begin
        if(!reset)
        out<=0;
    else
        out<=out + 1;                    //计数
    end
endmodule
```

程序说明：

本例只要 reset=0，计数器就清零，与时钟无关，为异步清零计数器。

5) 扭环形计数器

扭环形计数器即约翰逊计数器。

Verilog 语言描述：

```
//例 1.1.13
    module johnson(clk,clr,out);
    input clk,clr;
    output[3:0] out;
    reg[3:0] out;
    always @(posedge clk or negedge clr)
    begin
        if (!clr) out<=4'h0;
```

需要全本请访问 www.ertongbook.com