

第 1 步： 5 分

实验目的：

- (1) 学习 Vivado 软件中基于 Verilog 语言创建项目并下载使用的方法。
- (2) 初步学习 Verilog 语言的结构和语法。
- (3) 初步了解 Verilog 语言描述组合逻辑电路的语法。

第 2 步： 5 分

实验原理：

(XC7A35T-1CSG324C)具有大容量高性能等特点，能实现较复杂的数字逻辑设计。在 FPGA 内可以构建 MicroBlaze 处理器系统，可进行 SoC 设计。该平台拥有丰富的外设，以及灵活的通用扩展接口。

Verilog 语言基本元素及规则

1. 间隔符 Verilog 的间隔符主要起分隔文本的作用，可以使文本错落有致，便于阅读与修改。

间隔符包括空格符 (\b)、TAB 键 (\t)、换行符 (\n) 及换页符。

2. 注释符

注释只是为了改善程序的可读性，在编译时不起作用。

3. 标识符和关键词

标识符：给对象（如模块名、电路的输入与输出端口、变量等）取名所用的字符串。以英文字母或下划线开始，如，clk、counter8、_net、bus_A。

关键词：是 Verilog 语言本身规定的特殊字符串，用来定义语言的结构。例如，module、endmodule、input、output、wire、reg、and 等都是关键词。关键词都是小写，关键词不能作为标识符使用。

4. 逻辑值集合

为了表示数字逻辑电路的逻辑状态，Verilog 语言规定了 4 种基本的逻辑值。分别为 0—逻辑假，1—逻辑真，x 或 X—不确定，z 或 Z—高阻态。

5. 常量及其表示 Verilog 语言的常量包括整数型常量和实数型常量两大类型。其中整数类型常量有多种表示方法，包括十进制数表示，用于表示有符号常量，例如 30，-2 等；带基数形式的表示，格式为：<位宽><基数符号><数值>，例如 4'b1011、5'o37、8'hE3，8'b1001_0011 等。

6. 字符串

字符串是双撇号内的字符序列

Verilog 语言功能定义

1. 用 assign 语句 assign a = b & c; //二输入与门

2. 用实例元件 and and_inst(q,a,b)

其中 and 为设计库中已存在的实例元件名称，and_inst 为本次实例化的具体名称。q，a，b 分别为与 and 这一元件所对应连接的端口信号。3. 用 always 块 always 块既可用于描述组合逻辑也可描述时序逻辑，边沿触发的 always 块通常描述时序行为，电平触发的 always 块常用来描述组合逻辑行为。always 模块内，逻辑是按照指定的顺序执行的，两个

或更多的 always 模块也是同时执行的。

Verilog 语言变量种类

1、线网类型:

指输出始终根据输入的变化而更新其值的变量,它一般指的是硬件电路中的各种物理连接。

2、寄存器型:

寄存器型变量对应的是具有状态保持作用的电路等元件,如触发器寄存器。寄存器型变量 只能在 initial 或 always 内部被赋值。

第3步: 8分

预习题

数字电子技术实验预习						
实验二 使用 FPGA 实现组合逻辑电路						
专业-班级: <u>自动化131</u>		学号: <u>210320111</u>		姓名: <u>吕家昊</u>		
实验检查记录表:						
预习	一	二	三	四	必做完成时间	五(选做)
预习已学	预习已学	预习已学	预习已学	预习已学	16:28	预习已学

1、填空: 本次实验使用的 EGO1 实验板搭载的 FPGA 芯片是 xc7a35t-1csg324c

2、选择: 实验板上通用按键 S0~S4 默认状态是 【A】, 按下是 【B】。
A 低电平 B 高电平

3、填空: 实验板上拨码开关 SW7 对应 FPGA 管脚是 P5, 按键 S1 对应管脚是 R17, LED 灯 LD2_0 对应管脚是 K2。

4、填空: 硬件描述语言是一种 用形式化方法描述逻辑电路和系统的 语言。

5、填空: verilog 语言的标识符必须以 英文字母或下划线 开始。

6、填空: verilog 语言的 4 种基本逻辑值为 逻辑假(0)、逻辑真(1)、不确定(X)、高阻态(Z)

7、填空: verilog 语言的基本单元是 模块, 由两部分组成, 一部分描述 接口, 一部分描述 逻辑功能。

8、问答: verilog 语言功能描述有哪三种方式? 并请写出对应的示例语句。
结构化描述 例: and U3(out_Y0, EN, not-A1, not-A0) 表示与门 out_Y0 对应连接三输入端。
语言数据流描述: 例: assign out_Y0 = EN ? (in_A1, in_A0) == 2'b00 : 0 表示 EN=1 时若 in_A0, in_A1 均为 0 时 out_Y0 为 1, 否则为 0
行为级描述: if (EN == 0) {out_Y3 = reg, out_Y2 = reg, out_Y1 = reg, out_Y0 = reg} = 4'b0000;

9、问答: verilog 语言变量有哪两种类型? 分别可能会被综合为电路中的什么元件?
wire (线网类型): 硬件电路中各种物理连接 表示根据输入, 直接描述模块行为功能, 并存储在 reg 中
寄存器类: 具有状态保持作用的元件

10、问答: 本实验的示例约束文件中对电路进行了哪三种约束?
配置输入输出管脚, 电压标准, 未使用管脚非上拉非下拉。

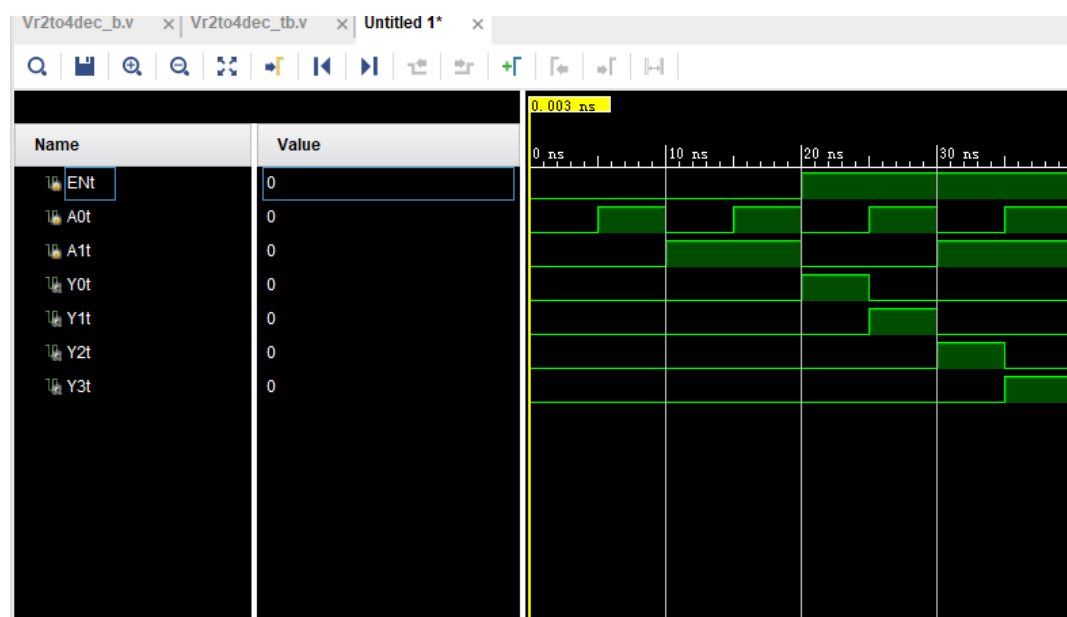
第5步：5分

2.5.1 使用 FPGA 设计实现一个 2-4 译码器

简述实验现象。

当 EN=0 时，输出引脚均为低电平；EN=1 时，根据输入引脚 A1、A0 电平情况，对应的

其中一个输出引脚 Yn 为高电平，其他输出为低电平。



```
set_property PACKAGE_PIN M4 [get_ports EN];
set_property PACKAGE_PIN N4 [get_ports in_A1];
set_property PACKAGE_PIN R1 [get_ports in_A0];
set_property PACKAGE_PIN K2 [get_ports out_Y0];
set_property PACKAGE_PIN J2 [get_ports out_Y1];
set_property PACKAGE_PIN J3 [get_ports out_Y2];
set_property PACKAGE_PIN H4 [get_ports out_Y3];
set_property IOSTANDARD LVCMOS33 [get_ports EN];
set_property IOSTANDARD LVCMOS33 [get_ports in_A1];
set_property IOSTANDARD LVCMOS33 [get_ports in_A0];
set_property IOSTANDARD LVCMOS33 [get_ports out_Y0];
set_property IOSTANDARD LVCMOS33 [get_ports out_Y1];
set_property IOSTANDARD LVCMOS33 [get_ports out_Y2];
set_property IOSTANDARD LVCMOS33 [get_ports out_Y3];
set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design];
```

```

module Vr2to4dec_tb(

);
  reg ENt, A0t, A1t;
  wire Y0t, Y1t, Y2t, Y3t;
  Vr2to4dec_b UUT(
    .EN(ENt), .in_A0(A0t), .in_A1(A1t),
    .out_Y0(Y0t), .out_Y1(Y1t), .out_Y2(Y2t), .out_Y3(Y3t));
  initial
    begin
      {ENt, A1t, A0t} = 3'b000;
      #5 {ENt, A1t, A0t} = 3'b001;
      #5 {ENt, A1t, A0t} = 3'b010;
      #5 {ENt, A1t, A0t} = 3'b011;
      #5 {ENt, A1t, A0t} = 3'b100;
      #5 {ENt, A1t, A0t} = 3'b101;
      #5 {ENt, A1t, A0t} = 3'b110;
      #5 {ENt, A1t, A0t} = 3'b111;
      #5 {ENt, A1t, A0t} = 3'b000;
      #10 $finish;
    end
endmodule

```

```

module Vr2to4dec_b(
  input EN, input in_A0, input in_A1,
  output out_Y0, output out_Y1, output out_Y2, output out_Y3
);
  reg out_Y0_reg, out_Y1_reg, out_Y2_reg, out_Y3_reg;
  always@ *
    begin
      if (EN==0)
        {out_Y3_reg, out_Y2_reg, out_Y1_reg, out_Y0_reg} = 4'b0000;
      else
        case({in_A1, in_A0})
          2'b00: {out_Y3_reg, out_Y2_reg, out_Y1_reg, out_Y0_reg} = 4'b0001;
          2'b01: {out_Y3_reg, out_Y2_reg, out_Y1_reg, out_Y0_reg} = 4'b0010;
          2'b10: {out_Y3_reg, out_Y2_reg, out_Y1_reg, out_Y0_reg} = 4'b0100;
          2'b11: {out_Y3_reg, out_Y2_reg, out_Y1_reg, out_Y0_reg} = 4'b1000;
          default: {out_Y3_reg, out_Y2_reg, out_Y1_reg, out_Y0_reg} = 4'b0000;
        endcase
      end
      assign out_Y0 = out_Y0_reg;
      assign out_Y1 = out_Y1_reg;
      assign out_Y2 = out_Y2_reg;
      assign out_Y3 = out_Y3_reg;
    end
endmodule

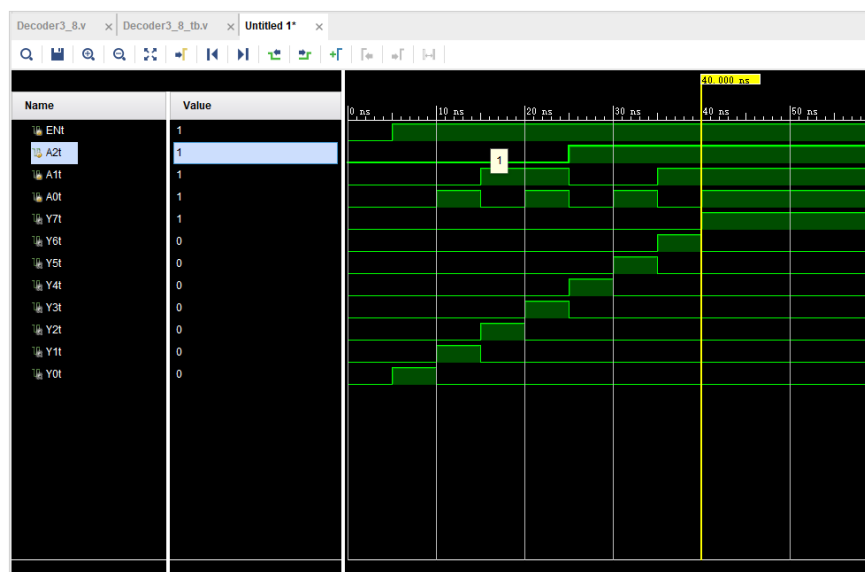
```

第 6 步： 15 分

2.5.2 使用 FPGA 设计实现一个 3-8 译码器

简述实验现象。

当 EN=0 时，输出引脚均为低电平；EN=1 时，根据输入引脚 A2~A0 电平情况，对应的其中一个输出引脚 Yn 为高电平，其他输出为低电平。



```
1 set_property PACKAGE_PIN P5 [get_ports EN];
2 set_property PACKAGE_PIN M4 [get_ports in_A2];
3 set_property PACKAGE_PIN N4 [get_ports in_A1];
4 set_property PACKAGE_PIN R1 [get_ports in_A0];
5 set_property PACKAGE_PIN K1 [get_ports out_D7];
6 set_property PACKAGE_PIN H6 [get_ports out_D6];
7 set_property PACKAGE_PIN H5 [get_ports out_D5];
8 set_property PACKAGE_PIN J5 [get_ports out_D4];
9 set_property PACKAGE_PIN K6 [get_ports out_D3];
10 set_property PACKAGE_PIN L1 [get_ports out_D2];
11 set_property PACKAGE_PIN M1 [get_ports out_D1];
12 set_property PACKAGE_PIN K3 [get_ports out_D0];
13
14 set_property IOSTANDARD LVCMOS33 [get_ports EN];
15 set_property IOSTANDARD LVCMOS33 [get_ports in_A2];
16 set_property IOSTANDARD LVCMOS33 [get_ports in_A1];
17 set_property IOSTANDARD LVCMOS33 [get_ports in_A0];
18 set_property IOSTANDARD LVCMOS33 [get_ports out_D7];
19 set_property IOSTANDARD LVCMOS33 [get_ports out_D6];
20 set_property IOSTANDARD LVCMOS33 [get_ports out_D5];
21 set_property IOSTANDARD LVCMOS33 [get_ports out_D4];
22 set_property IOSTANDARD LVCMOS33 [get_ports out_D3];
23 set_property IOSTANDARD LVCMOS33 [get_ports out_D2];
24 set_property IOSTANDARD LVCMOS33 [get_ports out_D1];
25 set_property IOSTANDARD LVCMOS33 [get_ports out_D0];
~
```

```

module Decoder3_8(
    input EN, input in_A2, input in_A1, input in_A0,
    output out_D7, output out_D6, output out_D5, output out_D4,
    output out_D3, output out_D2, output out_D1, output out_D0
);

    reg [7:0] temp;
    always@ *
    begin
        if(EN == 0)
            temp = 8'b00000000;
        else
            case({in_A2, in_A1, in_A0})
                3'b000:temp = 8'b00000001;
                3'b001:temp = 8'b00000010;
                3'b010:temp = 8'b00000100;
                3'b011:temp = 8'b00001000;
                3'b100:temp = 8'b00010000;
                3'b101:temp = 8'b00100000;
                3'b110:temp = 8'b01000000;
                3'b111:temp = 8'b10000000;
                default:temp = 8'b00000000;
            endcase
        end
        assign {out_D7, out_D6, out_D5, out_D4, out_D3, out_D2, out_D1, out_D0} = temp;
    endmodule

```

```

23 module Decoder3_8_tb();
24     reg ENt, A2t, A1t, A0t;
25     wire Y7t, Y6t, Y5t, Y4t, Y3t, Y2t, Y1t, Y0t;
26
27     Decoder3_8 uut(
28         .EN(ENt),
29         .in_A2(A2t), .in_A1(A1t), .in_A0(A0t),
30         .out_D7(Y7t), .out_D6(Y6t), .out_D5(Y5t), .out_D4(Y4t),
31         .out_D3(Y3t), .out_D2(Y2t), .out_D1(Y1t), .out_D0(Y0t));
32
33     initial
34     begin
35         {ENt, A2t, A1t, A0t} = 4'b0000;
36         #5 {ENt, A2t, A1t, A0t} = 4'b1000;
37         #5 {ENt, A2t, A1t, A0t} = 4'b1001;
38         #5 {ENt, A2t, A1t, A0t} = 4'b1010;
39         #5 {ENt, A2t, A1t, A0t} = 4'b1011;
40         #5 {ENt, A2t, A1t, A0t} = 4'b1100;
41         #5 {ENt, A2t, A1t, A0t} = 4'b1101;
42         #5 {ENt, A2t, A1t, A0t} = 4'b1110;
43         #5 {ENt, A2t, A1t, A0t} = 4'b1111;
44         #10 $finish;
45     end
46 endmodule
47

```

第 7 步： 20 分

2.5.3 设计一位全加器

简述实验现象。

当 CI, A, B 中一个或三个输入为高电平时，S 输出高电平，否则输出低电平

当 CI, A, B 中两个或三个输出为高电平时，CO 输出高电平，否则输出低电平

```
set_property PACKAGE_PIN M4 [get_ports CI];
set_property PACKAGE_PIN N4 [get_ports A];
set_property PACKAGE_PIN R1 [get_ports B];
set_property PACKAGE_PIN K2 [get_ports S];
set_property PACKAGE_PIN J2 [get_ports CO];
set_property IOSTANDARD LVCMOS33 [get_ports CI];
set_property IOSTANDARD LVCMOS33 [get_ports A];
set_property IOSTANDARD LVCMOS33 [get_ports B];
set_property IOSTANDARD LVCMOS33 [get_ports S];
set_property IOSTANDARD LVCMOS33 [get_ports CO];
set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design];
```

```
module Add_1Digit(
    input CI, A, B,
    output S, CO
);

    assign S = CI ^ A ^ B;
    assign CO = (A & B) | (A & CI) | (B & CI);
endmodule
```

第 8 步： 20 分

2.5.4 设计一个优先报警器

报警器电路功能如表 2.8，其中，X 表示任意值。

当 I0~I3 均为低电平时，Y0~Y2 为低电平

I3 为高电平时，Y0~Y2 为高电平

I3 为低电平，I2 为高电平时，Y0, Y1 为高电平，Y2 为低电平

I3, I2 为低电平，I1 为高电平时，Y0, Y2 为高电平，Y1 为低电平

I3, I2, I1 为低电平，I0 为高电平时，Y0 为高电平，Y1, Y2 为低电平

```
1  set_property PACKAGE_PIN R1 [get_ports I0];
2  set_property PACKAGE_PIN N4 [get_ports I1];
3  set_property PACKAGE_PIN M4 [get_ports I2];
4  set_property PACKAGE_PIN R2 [get_ports I3];
5  set_property PACKAGE_PIN K2 [get_ports Y0];
6  set_property PACKAGE_PIN J2 [get_ports Y1];
7  set_property PACKAGE_PIN J3 [get_ports Y2];
8  set_property IOSTANDARD LVCMOS33 [get_ports I0];
9  set_property IOSTANDARD LVCMOS33 [get_ports I1];
10 set_property IOSTANDARD LVCMOS33 [get_ports I2];
11 set_property IOSTANDARD LVCMOS33 [get_ports I3];
12 set_property IOSTANDARD LVCMOS33 [get_ports Y0];
13 set_property IOSTANDARD LVCMOS33 [get_ports Y1];
14 set_property IOSTANDARD LVCMOS33 [get_ports Y2];
15 set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design];

3  module Alarmer(
4      input I0, I1, I2, I3,
5      output Y0, Y1, Y2
6  );
7      reg [2:0] temp;
8      always@ *
9      begin
10         if({I0, I1, I2, I3} == 4'b0000) temp = 3'b000;
11         else if(I3 == 1) temp = 3'b111;
12         else if(I2 == 1) temp = 3'b110;
13         else if(I1 == 1) temp = 3'b101;
14         else if(I0 == 1) temp = 3'b100;
15         end
16         assign {Y0, Y1, Y2} = temp;
17     endmodule
```


第 9 步： 10 分

2.5.5* (选做) 设计减法运算显示电路

简述实验现象。

当 SW7~SW4 输入被减数大于等于 SW3~SW0 输入减数时，数码管显示两数之差，

K2 灯灭

当被减数小于减数时，数码管显示两数之差的绝对值，K2 灯亮

(K2 用于表示负号)

```
1  set_property PACKAGE_PIN P5 [get_ports in13];
2  set_property PACKAGE_PIN P4 [get_ports in12];
3  set_property PACKAGE_PIN P3 [get_ports in11];
4  set_property PACKAGE_PIN P2 [get_ports in10];
5  set_property PACKAGE_PIN R2 [get_ports in03];
6  set_property PACKAGE_PIN M4 [get_ports in02];
7  set_property PACKAGE_PIN N4 [get_ports in01];
8  set_property PACKAGE_PIN R1 [get_ports in00];
9  set_property PACKAGE_PIN B4 [get_ports led11];
10 set_property PACKAGE_PIN A4 [get_ports led12];
11 set_property PACKAGE_PIN A3 [get_ports led13];
12 set_property PACKAGE_PIN B1 [get_ports led14];
13 set_property PACKAGE_PIN A1 [get_ports led15];
14 set_property PACKAGE_PIN B3 [get_ports led16];
15 set_property PACKAGE_PIN B2 [get_ports led17];
16 set_property PACKAGE_PIN G2 [get_ports en1];
17 set_property PACKAGE_PIN D4 [get_ports led21];
18 set_property PACKAGE_PIN E3 [get_ports led22];
19 set_property PACKAGE_PIN D3 [get_ports led23];
20 set_property PACKAGE_PIN F4 [get_ports led24];
21 set_property PACKAGE_PIN F3 [get_ports led25];
22 set_property PACKAGE_PIN E2 [get_ports led26];
23 set_property PACKAGE_PIN D2 [get_ports led27];
24 set_property PACKAGE_PIN G1 [get_ports en2];
25 set_property PACKAGE_PIN K2 [get_ports minus];
~
```

```

27 set_property IOSTANDARD LVCMOS33 [get_ports in13];
28 set_property IOSTANDARD LVCMOS33 [get_ports in12];
29 set_property IOSTANDARD LVCMOS33 [get_ports in11];
30 set_property IOSTANDARD LVCMOS33 [get_ports in10];
31 set_property IOSTANDARD LVCMOS33 [get_ports in03];
32 set_property IOSTANDARD LVCMOS33 [get_ports in02];
33 set_property IOSTANDARD LVCMOS33 [get_ports in01];
34 set_property IOSTANDARD LVCMOS33 [get_ports in00];
35 set_property IOSTANDARD LVCMOS33 [get_ports led11];
36 set_property IOSTANDARD LVCMOS33 [get_ports led12];
37 set_property IOSTANDARD LVCMOS33 [get_ports led13];
38 set_property IOSTANDARD LVCMOS33 [get_ports led14];
39 set_property IOSTANDARD LVCMOS33 [get_ports led15];
40 set_property IOSTANDARD LVCMOS33 [get_ports led16];
41 set_property IOSTANDARD LVCMOS33 [get_ports led17];
42 set_property IOSTANDARD LVCMOS33 [get_ports en1];
43 set_property IOSTANDARD LVCMOS33 [get_ports led21];
44 set_property IOSTANDARD LVCMOS33 [get_ports led22];
45 set_property IOSTANDARD LVCMOS33 [get_ports led23];
46 set_property IOSTANDARD LVCMOS33 [get_ports led24];
47 set_property IOSTANDARD LVCMOS33 [get_ports led25];
48 set_property IOSTANDARD LVCMOS33 [get_ports led26];
49 set_property IOSTANDARD LVCMOS33 [get_ports led27];
50 set_property IOSTANDARD LVCMOS33 [get_ports en2];
51 set_property IOSTANDARD LVCMOS33 [get_ports minus];
52
53 set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design];

```

```

23 module Minus(
24     input in13, in12, in11, in10, in03, in02, in01, in00, //in1为被减数 in0为减数
25     output led11, led12, led13, led14, led15, led16, led17, led21, led22, led23, led24, led25, led26, led27, //数码管
26     output en1, en2, minus
27 );
28     reg [3:0] result;
29     reg minus_flag;
30     reg [6:0] led1, led2;
31     always@ *
32     begin
33         if({in13, in12, in11, in10} >= {in03, in02, in01, in00})
34         begin
35             result = {in13, in12, in11, in10} - {in03, in02, in01, in00};
36             minus_flag = 0;
37         end
38         else
39         begin
40             result = {in03, in02, in01, in00} - {in13, in12, in11, in10};
41             minus_flag = 1;
42         end
43         case(result)
44             0: {led2, led1} = 14'b11111101111110;
45             1: {led2, led1} = 14'b11111100110000;
46             2: {led2, led1} = 14'b11111101101101;
47             3: {led2, led1} = 14'b11111101111001;
48             4: {led2, led1} = 14'b11111100110011;
49             5: {led2, led1} = 14'b11111101011011;
50             6: {led2, led1} = 14'b11111101011111;
51             7: {led2, led1} = 14'b11111101110000;
52             8: {led2, led1} = 14'b11111101111111;
53             9: {led2, led1} = 14'b11111101111011;

```

```

44:         0: {led2, led1} = 14'b11111101111110;
45:         1: {led2, led1} = 14'b11111100110000;
46:         2: {led2, led1} = 14'b11111101101101;
47:         3: {led2, led1} = 14'b11111101111001;
48:         4: {led2, led1} = 14'b11111100110011;
49:         5: {led2, led1} = 14'b11111101011011;
50:         6: {led2, led1} = 14'b11111101011111;
51:         7: {led2, led1} = 14'b11111101110000;
52:         8: {led2, led1} = 14'b11111101111111;
53:         9: {led2, led1} = 14'b11111101111011;
54:         10: {led2, led1} = 14'b01100001111110;
55:         11: {led2, led1} = 14'b01100000110000;
56:         12: {led2, led1} = 14'b01100001101101;
57:         13: {led2, led1} = 14'b01100001111001;
58:         14: {led2, led1} = 14'b01100000110011;
59:         15: {led2, led1} = 14'b01100001011011;
60:     endcase
61: end
62:     assign {en1, en2} = 3;
63:     assign {led11, led12, led13, led14, led15, led16, led17} = led2;
64:     assign {led21, led22, led23, led24, led25, led26, led27} = led1;
65:     assign minus = minus_flag;
66: endmodule
67:

```

实验思考

1.verilog 通常有哪几种方式描述逻辑电路？各有什么特点和适用场景？

结构化描述：使用实例化低层次模块的方法对整个电路的功能进行描述，或者直接调用 Verilog 内部预先定义的基本门级元件描述电路的结构，适用于描述由逻辑门直接组成的电路。

数据流描述：使用连续赋值语句(assign)对电路的逻辑功能进行描述，适用于对组合逻辑电路进行建模。

行为级描述：侧重于描述模块的行为功能，由 EDA 软件根据代码来综合出逻辑电路。

通常使用 always 与高级程序语句描述逻辑功能，较为直观，适用于实现较复杂的功能。

2. 在 vivado 软件中使用 verilog 语言描述实现数字电路时需要经过哪些步骤？

创建工程文件，编写设计文件、仿真文件与约束文件后，需要综合电路文件，然后进行布局布线，实现电路元件。此后生成比特流文件，并下载到板中运行。