**实验报告**

2019 年 4 月 5 日 成绩：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | | 王坤 | 学号 | | 17051225 | | 班级 | | 17052312 | |
| 专业 | | 计算机科学与技术 | | | 课程名称 | | 计算机组成原理 | | | |
| 任课老师 | | 章复嘉 | 指导老师 | | 章复嘉 | | 机位号 | |  | |
| 实验序号 | | 实验三 | 实验名称 | | 多功能ALU | | | | | |
| 实验时间 | | 2018.4.4 | 实验地点 | | 1-225 | | 实验设备号 | |  | |
| **一、实验目的与要求** | | | | | | | | | | |
| 1. 实验目的：学习多功能ALU的工作原理，掌握运算器的设计方法   掌握运用Verilog HDL语言进行行为描述与建模的技巧和方法   1. 实验要求：要求设计一个具有8种运算功能的32位ALU，并能够产生运算结果的标志：   结果为零标志ZF  溢出标志OF  编写顶层模块：用于验证模块的正确性； | | | | | | | | | | |
| **二、实验设计与程序代码** | | | | | | | | | | |
| 1. 模块设计说明   本程序使用顶层模块top用于验证模块的正确性,子模块ALU设计程序  本模块全部采用非阻塞语句赋值  本模块使用一个32位的输入,一个32位的输出,使用6个按钮进行功能复用  子模块ALU使用case语句进行功能的选择  /**/ALU模块**  ``timescale 1ns / 1ps  module ALU(OP,A,B,F,ZF,CF,OF,SF,PF);  parameter SIZE = 32;//运算位数  input [3:0] OP;//运算操作  input [SIZE:1] A;//左运算数  input [SIZE:1] B;//右运算数  output [SIZE:1] F;//运算结果  output ZF, //0标志位, 运算结果为0(全零)则置1, 否则置0  CF, //进借位标志位, 取最高位进位C,加法时C=1则CF=1表示有进位,减法时C=0则CF=1表示有借位  OF, //溢出标志位，对有符号数运算有意义，溢出则OF=1，否则为0  SF, //符号标志位，与F的最高位相同  PF; //奇偶标志位，F有奇数个1，则PF=1，否则为0  reg [SIZE:1] F;  reg C,ZF,CF,OF,SF,PF;//C为最高位进位  always@(\*)  begin  C=0;  case(OP)  4'b0000:begin F=A&B; end //按位与  4'b0001:begin F=A|B; end //按位或  4'b0010:begin F=A^B; end //按位异或  4'b0011:begin F=~(A|B); end //按位或非  4'b0100:begin {C,F}=A+B; end //加法  4'b0101:begin {C,F}=~B+A+1; end //减法  4'b0110:begin F=A<B; end //A<B则F=1，否则F=0  4'b0111:begin F=B<<A; end //将B左移A位  endcase  ZF = F==0;//F全为0，则ZF=1  CF = C; //进位借位标志  OF = A[SIZE]^B[SIZE]^F[SIZE]^C;//溢出标志公式  SF = F[SIZE];//符号标志,取F的最高位  PF = ~^F;//奇偶标志，F有奇数个1，则F=1；偶数个1，则F=0  end  endmodule  **顶层模块CPU**    `timescale 1ns / 1ps  module top(  SW, //使能开关  Input\_Data, //数据输入  Output\_Data //数据输出  );  parameter ADDR = 5;//地址位宽  parameter SIZE = 32;//数据位宽  //外部数据输入口  input [SIZE:1]Input\_Data;  //按钮  input [5:0] SW;  //led输出  output reg [SIZE:1] Output\_Data;  reg [SIZE:1]R\_Data\_A;//A端口  reg [SIZE:1]R\_Data\_B;//B端口    //ALU  reg [3:0] OP;//运算符编码  wire ZF,//零标志  CF,//进借位标志  OF,//溢出标志  SF,//符号标志  PF;//奇偶标志  wire [SIZE:1]ALU\_F;//ALU运算结果中间变量  //实例化ALU模块  ALU ALU\_Test(  .OP(OP),//运算符  .A(R\_Data\_A),//A操作数  .B(R\_Data\_B),//B操作数  .F(ALU\_F),//ALU\_F作为中间变量    .ZF(ZF),//零标志  .CF(CF),//进借位标志  .OF(OF),//溢出标志  .SF(SF),//符号标志  .PF(PF)//奇偶标志  );    always@(\*)  begin  if(SW[1])  begin  R\_Data\_A<= Input\_Data;  end  if(SW[2]) //保存寄存数据  begin  R\_Data\_B<= Input\_Data;  end  if(SW[3])  begin  OP <= Input\_Data[4:1]; ////保存操作符号  end  if(SW[4]) //打印云算结果  begin  Output\_Data<= ALU\_F; //运算结果    end  if(SW[5]) //打印云算结果  begin  Output\_Data<= {27'b0,ZF,CF,OF,SF,PF}; //运算标志符    end  end  endmodule | | | | | | | | | | |
| **三、实验仿真** | | | | | | | | | | |
| 1. 仿真代码   `timescale 1ns / 1ps  ////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date: 18:59:07 04/18/2019  // Design Name: top  // Module Name: E:/ISE/Term\_PC/alu/test.v  // Project Name: alu  // Target Device:  // Tool versions:  // Description:  //  // Verilog Test Fixture created by ISE for module: top  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  ////////////////////////////////////////////////////////////////////////////////  module test;  // Inputs  reg [5:0] SW;  reg [32:1] Input\_Data;  // Outputs  wire [32:1] Output\_Data;  // Instantiate the Unit Under Test (UUT)  top uut (  .SW(SW),  .Input\_Data(Input\_Data),  .Output\_Data(Output\_Data)  );  initial begin  // Initialize Inputs  SW = 0;  Input\_Data = 0;  // Wait 100 ns for global reset to finish  #10;  //按位与  Input\_Data = 32'h0001\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'h0001\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0000;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;    //按位或  Input\_Data = 32'h0001\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'h0001\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0001;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  //按位异或  Input\_Data = 32'h0001\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'h0001\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0002;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  //按位或非  Input\_Data = 32'h0001\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'h0001\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0003;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  //加法  Input\_Data = 32'hfff1\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'h0001\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0004;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  //减法  Input\_Data = 32'h0001\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'hfff1\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0005;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  //A<B则F=1，否则F=0  Input\_Data = 32'h0001\_ffff;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'h0001\_000f;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0006;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  //将B左移A位  Input\_Data = 32'h0000\_0004;SW[1] =1;#10;SW[1]=0;#10;  Input\_Data = 32'hffff\_ffff;SW[2] =1;#10;SW[2]=0;#10;  Input\_Data = 32'h0000\_0007;SW[3] =1;#10;SW[3]=0;#10;  SW[4] =1;#10;SW[4]=0;#10;  SW[5] =1;#10;SW[5]=0;#10;  end    endmodule   1. 仿真波形(限制于篇幅,波形图只是对单一运算进行了单一数据仿真,仿真的顺序为上面仿真文件所述)                    1. 仿真结果分析   本实验采用的数据所得到的结果和理论计算上是完全一致的  各种情况的数据如下(第一个为左操作数,第二个为右操作数,实际验证的操作数据远比这多,这里只是每种情况举一例) | | | | | | | | | | |
| **四、电路图** | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| **五、引脚配置（约束文件）** | | | | | | | | | | |
| NET "Input\_Data[31]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[30]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[29]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[28]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[27]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[26]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[25]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[24]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[23]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[22]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[21]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[20]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[19]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[18]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[17]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[16]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[15]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[14]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[13]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[12]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[11]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[10]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[9]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[8]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[7]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[6]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[5]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[4]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[3]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[2]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[1]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[32]" IOSTANDARD = LVCMOS18;  NET "Input\_Data[32]" LOC = T3;  NET "Input\_Data[31]" LOC = U3;  NET "Input\_Data[30]" LOC = T4;  NET "Input\_Data[29]" LOC = V3;  NET "Input\_Data[28]" LOC = V4;  NET "Input\_Data[27]" LOC = W4;  NET "Input\_Data[26]" LOC = Y4;  NET "Input\_Data[25]" LOC = Y6;  NET "Input\_Data[24]" LOC = W7;  NET "Input\_Data[23]" LOC = Y8;  NET "Input\_Data[22]" LOC = Y7;  NET "Input\_Data[21]" LOC = T1;  NET "Input\_Data[20]" LOC = U1;  NET "Input\_Data[19]" LOC = U2;  NET "Input\_Data[18]" LOC = W1;  NET "Input\_Data[17]" LOC = W2;  NET "Input\_Data[16]" LOC = Y1;  NET "Input\_Data[15]" LOC = AA1;  NET "Input\_Data[14]" LOC = V2;  NET "Input\_Data[13]" LOC = Y2;  NET "Input\_Data[12]" LOC = AB1;  NET "Input\_Data[11]" LOC = AB2;  NET "Input\_Data[10]" LOC = AB3;  NET "Input\_Data[9]" LOC = AB5;  NET "Input\_Data[8]" LOC = AA6;  NET "Input\_Data[7]" LOC = R2;  NET "Input\_Data[6]" LOC = R3;  NET "Input\_Data[5]" LOC = T6;  NET "Input\_Data[4]" LOC = R6;  NET "Input\_Data[3]" LOC = U7;  NET "Input\_Data[2]" LOC = AB7;  NET "Input\_Data[1]" LOC = AB8;  NET "Input\_Data[31]" PULLDOWN;  NET "Input\_Data[30]" PULLDOWN;  NET "Input\_Data[29]" PULLDOWN;  NET "Input\_Data[28]" PULLDOWN;  NET "Input\_Data[27]" PULLDOWN;  NET "Input\_Data[26]" PULLDOWN;  NET "Input\_Data[25]" PULLDOWN;  NET "Input\_Data[24]" PULLDOWN;  NET "Input\_Data[23]" PULLDOWN;  NET "Input\_Data[22]" PULLDOWN;  NET "Input\_Data[21]" PULLDOWN;  NET "Input\_Data[20]" PULLDOWN;  NET "Input\_Data[19]" PULLDOWN;  NET "Input\_Data[18]" PULLDOWN;  NET "Input\_Data[17]" PULLDOWN;  NET "Input\_Data[16]" PULLDOWN;  NET "Input\_Data[15]" PULLDOWN;  NET "Input\_Data[14]" PULLDOWN;  NET "Input\_Data[13]" PULLDOWN;  NET "Input\_Data[12]" PULLDOWN;  NET "Input\_Data[11]" PULLDOWN;  NET "Input\_Data[10]" PULLDOWN;  NET "Input\_Data[9]" PULLDOWN;  NET "Input\_Data[8]" PULLDOWN;  NET "Input\_Data[7]" PULLDOWN;  NET "Input\_Data[6]" PULLDOWN;  NET "Input\_Data[5]" PULLDOWN;  NET "Input\_Data[4]" PULLDOWN;  NET "Input\_Data[3]" PULLDOWN;  NET "Input\_Data[2]" PULLDOWN;  NET "Input\_Data[1]" PULLDOWN;  NET "Input\_Data[32]" PULLDOWN;  NET "Output\_Data[31]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[30]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[29]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[28]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[27]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[26]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[25]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[24]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[23]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[22]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[21]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[20]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[19]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[18]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[17]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[16]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[15]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[14]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[13]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[12]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[11]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[10]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[9]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[8]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[7]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[6]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[5]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[4]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[3]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[2]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[1]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[32]" IOSTANDARD = LVCMOS18;  NET "Output\_Data[32]" LOC = R1;  NET "Output\_Data[31]" LOC = P2;  NET "Output\_Data[30]" LOC = P1;  NET "Output\_Data[29]" LOC = N2;  NET "Output\_Data[28]" LOC = M1;  NET "Output\_Data[27]" LOC = M2;  NET "Output\_Data[26]" LOC = L1;  NET "Output\_Data[25]" LOC = J2;  NET "Output\_Data[24]" LOC = G1;  NET "Output\_Data[23]" LOC = E1;  NET "Output\_Data[22]" LOC = D2;  NET "Output\_Data[21]" LOC = A1;  NET "Output\_Data[20]" LOC = L3;  NET "Output\_Data[19]" LOC = G3;  NET "Output\_Data[18]" LOC = K4;  NET "Output\_Data[17]" LOC = G4;  NET "Output\_Data[16]" LOC = K1;  NET "Output\_Data[15]" LOC = J1;  NET "Output\_Data[14]" LOC = H2;  NET "Output\_Data[13]" LOC = G2;  NET "Output\_Data[12]" LOC = F1;  NET "Output\_Data[11]" LOC = E2;  NET "Output\_Data[10]" LOC = D1;  NET "Output\_Data[9]" LOC = B1;  NET "Output\_Data[8]" LOC = B2;  NET "Output\_Data[7]" LOC = N3;  NET "Output\_Data[6]" LOC = M3;  NET "Output\_Data[5]" LOC = K3;  NET "Output\_Data[4]" LOC = H3;  NET "Output\_Data[3]" LOC = N4;  NET "Output\_Data[2]" LOC = L4;  NET "Output\_Data[1]" LOC = J4;  NET "SW[4]" IOSTANDARD = LVCMOS18;  NET "SW[3]" IOSTANDARD = LVCMOS18;  NET "SW[2]" IOSTANDARD = LVCMOS18;  NET "SW[1]" IOSTANDARD = LVCMOS18;  NET "SW[0]" IOSTANDARD = LVCMOS18;  NET "SW[4]" LOC = AA4;  NET "SW[3]" LOC = AB6;  NET "SW[2]" LOC = T5;  NET "SW[1]" LOC = V8;  NET "SW[0]" LOC = AA8;  NET "SW[0]" CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "SW[1]" CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "SW[4]" CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "SW[5]" CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "SW[3]" CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "SW[2]" CLOCK\_DEDICATED\_ROUTE = FALSE;  # PlanAhead Generated physical constraints  NET "SW[5]" LOC = R4;  # PlanAhead Generated IO constraints  NET "SW[5]" IOSTANDARD = LVCMOS18; | | | | | | | | | | |
| **六、思考与探索** | | | | | | | | | | |
|  | **A=h’ffff\_ffff**  **B=h’0001\_0001** | | | **A=h’0001\_0001**  **B=h’ffff\_ffff** | | **A=h’ffff\_0001**  **B=h’ffff\_0001** | | **A=h’0f0f\_0f0f**  **B=h’f0f0\_f0f0** | |
| A·B | F= h’0001\_0001  ZF= 0 OF=0 | | | F= h’0001\_0001  ZF=0 OF=0 | | F= h’ffff\_0001  ZF= 0 OF=0 | | F= h’0000\_0000  ZF= 1 OF=0 | |
| A+B | F= h’ffff\_ffff  ZF= 0 OF=0 | | | F= h’ffff\_ffff  ZF= 0 OF=0 | | F= h’ffff\_0001  ZF= 0 OF=0 | | F= h’ffff\_ffff  ZF= 0 OF=0 | |
| A⊕B | F= h’fffe\_fffe  ZF= 00 OF=0 | | | F= h’fffe\_fffe  ZF= 00 OF=0 | | F= h’0000\_0000  ZF= 1 OF=0 | | F= h’ffff\_ffff  ZF= 0 OF=0 | |
|  | F= h’0000\_0000  ZF= 1 OF=0 | | | F= h’0000\_0000  ZF= 1 OF=0 | | F= h’0000\_fff0  ZF= 0 OF=0 | | F= h’0000\_0000  ZF= 1 OF=0 | |
| A加B | F= h’0001\_0000  ZF= 0 OF=0 | | | F= h’0001\_0000  ZF= 0 OF=0 | | F= h’fffe\_0002  ZF= 0 OF=0 | | F= h’ffff\_ffff  ZF= 0 OF=0 | |
| A减B | F= h’fffe\_fffe  ZF= 0 OF=0 | | | F= h’0001\_0002  ZF= 0 OF=0 | | F= h’0000\_0000  ZF= 1 OF=0 | | F= h’1e1e\_1e1f  ZF= 0 OF=0 | |
| A<B置1 | F= h’0000\_0000  ZF=1 OF=0 | | | F= h’0000\_0001  ZF= 0 OF=0 | | F= h’0000\_0000  ZF= 1 OF=0 | | F= h’0000\_0001  ZF= 0 OF=0 | |
| B<<A | F= h’fffe\_0000  ZF= 0 OF=0 | | | F= h’0000\_0000  ZF= 1 OF=0 | | F= h’0000\_0000  ZF= 1 OF=0 | | F= h’\_  ZF= 0 OF=0 | |
| 1. 实验结果记录：见上表      1. 实验结论：实验的大小比较功能是对于无符号数而言的   实验的减法可以通过加法进行     1. 问题与解决方案：   这次的实验主要遇到了两个问题:  问题一: 由于板卡只有32位输入和输出,怎么样才能实现操作数和操作功能的输入?  解决方案: 本次主要是使用了3个按钮去实现这个功能,在撰写代码之前,的确是将两个操作数 和操作符全部写入了input,,但是在配置管脚的时候,却提示该管脚已被使用,于是将三个输入合一,采用按钮去选择输入,将三个输入复用了.但是这种做法带来了一个问题,也就是下面的这个问题!  问题二: 紧接着上面的叙述,在将三个输入复用后,发现进位符 ‘C’无法使用了?什么意思呢? 本来 在进行加法或者减法的时候, 表示式应该这么写: {C,F} <= temp\_a + temp\_b/{C,F} <= ~temp\_b+temp\_a+1;但是在仿真的时候,发现在不够减或者产生进位的这种情况,无法置入操作数,不知道是什么原因,  解决方案: 这个问题困惑了我很久,折腾了好几天都没有找到原因,然后,我采用了另一种方式去表达C,就是再使用一个’reg tempres[33:1]’,然后,在每次加法或者减法的时候,把原句该如下:  加法:F<=temp\_a +temp\_b;tempres<=temp\_a+tgemp\_b;  减法:F <= ~temp\_b+temp\_a+1; tempres<= ~temp\_b+temp\_a+1;  这样一来间接的使用了一个tempres[33]去代替这’C’,这样写就不会出现上面提到的问题,至于为什么会出现那种问题,我也没找到原因.  PS:上面的两个问题是我在没有编写顶级模块的时候出现的,然后,我使用了顶级模块去验证子模块,就没有出现上面的额两个问题了   1. 思考题：   逻辑右移和逻辑左移是一样的,整体进行移动,最左边补0,而算术右移则需要按照最高位进行部位,最高位为0则补0为1 则补1,不改变符号位,所以可以采用下列语句  Temp<=A[max]  A<=A>>index;  if(A[max]) begin A<={index’b1,A[max-index:min]}; end  至于为什么计算机里面没有算术左移,我猜测,可能是因为算术移位对应的是有符号数,而有符号数的符号位在最高位,移位无法保证其符号位不改变! | | | | | | | | | | |