《数字逻辑》实验报告

姓名		年级	2022 级
学号		专业、班级	计算机科学与技术（卓越） 1 班

实验名称	实验十三 流水线加法器		
实验时间	2023.11.9	实验地点	计算机机房 DS1410
实验成绩		实验性质	☐验证性 ☑设计性 ☐综合性

教师评价：

☐算法/实验过程正确； ☐源程序/实验内容提交 ☐程序结构/实验步骤合理；

☐实验结果正确； ☐语法、语义正确； ☐报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

1、掌握寄存器的设计与应用。

2、学习和掌握流水线加法器的设计与实现。

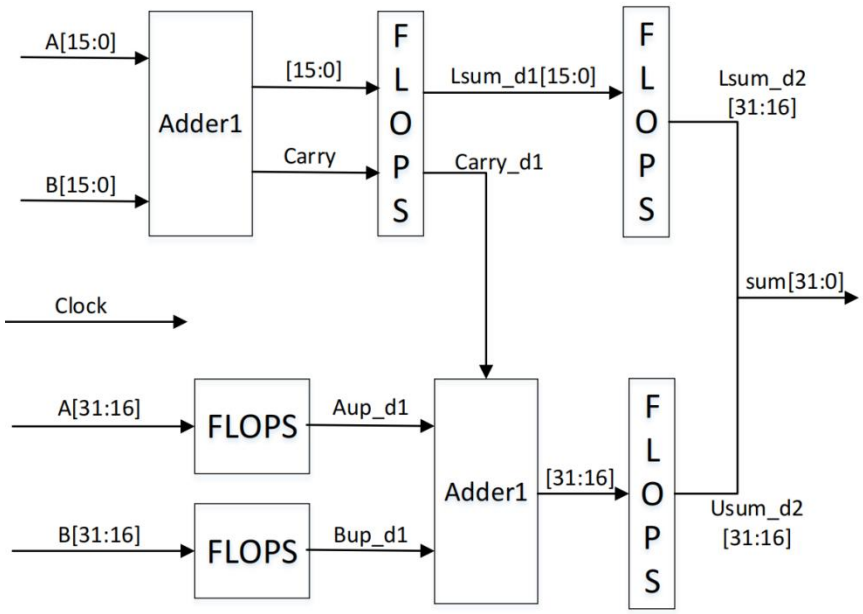
二、实验项目内容

1、设计并实现一个 32 位 2 级流水线加法器，并下板验证。

2、主要从 RTL 电路分析、仿真波形、开发板资源使用情况等方面与实验六中的加法器和系统自带“+”加法器进行比较。

三、实验设计（实验原理、真值表、原理图等）

32 位的 2 级流水线加法器使用两个 16 位加法器，其中第一个加法器执行低 16 位的加法，而第二个加法器执行高 32 位的加法，电路结构图如下图所示（供参考）。



具体的实验原理为，两个 16 位加减法器同时进行加减法，从而使原本串行的实验并行完成，提高了加减法的效率

四、实验过程或算法（关键步骤、核心代码注解等）

（1）代码模块详解

①顶层模块（调用三十二位加减法器，控制数码管的显示）

```
module topuse(
    input clk,
    input reset,
```

```
input [3:0] hex0, //第一个数码管显示的数字
input [3:0] hex1,
output reg [3:0] an,  //片选,使能端
output reg [7:0] sseg, //段选，最后数码管的输出
input c0,
output C32,
    output Gm,
    //进位产生标志（该输出为 1 时，不管 c0 真值情况，一定会进位）
    output Pm
    //进位传递标志（该输出为 1 时，需要 c0 为 1，才会进位）
);

        wire [32:1] hex3;

        parameter N = 18; //使用低 16 位对 50Mhz 时钟进行分频(50MHZ/2^16)
        reg [N-1:0] regN; //高两位作为控制信号，低 16 位为计数器，对时钟进
行分频
        reg [3:0] hex_in; //段选控制信号
        reg dp;
        wire dp_operator,dp_result;

        j32j a(
        .in1({28'b0,hex0}),
        .in2({28'b0,hex1}),
        .add_sub(c0),
        .Gm(Gm),
        .Pm(Pm),
        .sum(hex3),
        .c32(C32),
        .clk(clk),
        .rst_n(reset)
        );

        assign dp_operator=c0,
            dp_result=c0&&(~C32);
```

```
        always@(posedge clk, posedge reset)//低十六位开始计数，当计数到前两位时显示数码管
        begin
            if(reset)
                regN <= 0;
            else
                regN <= regN + 1;
        end

        always@ (*)
        begin
            case(regN[N-1:N-2])
            2'b00:begin
                an = 4'b0111; //选中第 1 个数码管
                hex_in = hex0; //数码管显示 hex0 输入的数字；
                dp = 0; //控制该数码管的小数点的亮灭
            end
            2'b01:begin
                an = 4'b1011; //选中第二个数码管
                hex_in = hex1;//数码管显示 hex1 输入的数字；
                dp = dp_operator;
            end
            2'b10:begin
                an = 4'b1110;//选中第三个数码管
                hex_in = hex3[4:1];//数码管显示 hex3 输入的数字；
                dp = dp_result;
            end
            default:
             begin
                an = 4'b1110;//选中第四个数码管
                hex_in = hex3[4:1];//数码管显示的数字由 hex_in 控制，显示hex3 输入的数字；
                dp = dp_result;
```

```
            end


            endcase


        end
        always@ (*)
        begin
            case(hex_in)//数码管的显示
                    4'h0: sseg[6:0] = 7'b0000001; //共阳极数码管
                    4'h1: sseg[6:0] = 7'b1001111;
                    4'h2: sseg[6:0] = 7'b0010010;
                    4'h3: sseg[6:0] = 7'b0000110;
                    4'h4: sseg[6:0] = 7'b1001100;
                    4'h5: sseg[6:0] = 7'b0100100;
                    4'h6: sseg[6:0] = 7'b0100000;
                    4'h7: sseg[6:0] = 7'b0001111;
                    4'h8: sseg[6:0] = 7'b0000000;
                    4'h9: sseg[6:0] = 7'b0001100;
                    4'ha: sseg[6:0] = 7'b0001000;
                    4'hb: sseg[6:0] = 7'b1100000;
                    4'hc: sseg[6:0] = 7'b1110010;
                    4'hd: sseg[6:0] = 7'b1000010;
                    4'he: sseg[6:0] = 7'b0110000;
                    4'hf: sseg[6:0] = 7'b0111000;
                    default: sseg[6:0] = 7'b0111000;
            endcase
            sseg[7] = ~dp;
        end
endmodule
```

②32 位加减法器模块（调用三十二位加法器模块，完成**减法**时的输出）

```
module j32j(
          input [32:1]in1,
        input [32:1]in2,
```

```
                input add_sub,
                output Gm,
                output Pm,
                output [32:1]sum,
                output c32,
                input   clk,
                input   rst_n
);
wire [32:1] in1_exp;
wire [32:1] in2_exp;
wire cin;
wire [32:1]sum_temp1;
wire [32:1]sum_temp2;

assign in1_exp = {in1};
assign in2_exp = add_sub? ~{in2}: {in2};     //做减法时，需要对 in2 进行补码运算
assign cin = add_sub;                        //对 in2 进行 2 的补码运算

// 因为对输入进行了扩位,所以 WIDTH 需要加 1，在输出的时候会被截位，但对结果无影响
adder32 adder32(//调用 32 位加法器进行运算
   .A( in1_exp   ),
   .B( in2_exp   ),
   .c0( cin   ),
   .Gm(Gm),
   .Pm(Pm),
   .S( sum_temp1   ),
   .C32(c32),
   .clk(clk),
   .rst_n(rst_n)
);



adder32_clu a(//调用加法器取补码
   .A( ~sum_temp1   ),
```

```
    .B(  0   ),
    .c0( 1'b1  ),
    .Gm( ),
    .Pm( ),
    .S( sum_temp2  ),
    .C32( )
);


assign sum = (add_sub&(~c32))? {sum_temp2}: {sum_temp1};      //做减法时，若结果为负数，需要对 in2 进行补码运算


Endmodule
```

③32 位加法器模块（32 位加法器模块调用 32 位**流水线**加法器模块，32 位流水线加法器由两个并行的 16 位 CLA(carray lookahead unit)实现）

```
module adder32(A,B,c0,Gm,Pm,S,C32,clk,rst_n);
    input [32:1] A;
                input [32:1] B;
                input c0;
                output Gm;
                output Pm;
                output [32:1] S;
                output C32;
                input    clk;
                input    rst_n;


                wire px1,gx1,px2,gx2;
                wire c16;



                adder_pipeline_32bit_2level adder32_to_pipeline_32bit(//调用 32 位流水线加法器
                 .FinalSum(S),
                 .co(C32),
                 .A(A),
```

```
                    .B(B),
                  .ci(c0),
                  .clk(clk),
                  .reset(rst_n),

                  .px1(px1),
                  .gx1(gx1),
                  .px2(px2),
                  .gx2(gx2)
);

 assign//完成进位等的计算
       Gm = gx2 ^ (gx1 & px2) ^ (px1 & c0),
       Pm = px1 & px2;

endmodule

//adder_pipeline_32bit_2level 32 位流水线模块代码
module adder_pipeline_32bit_2level(A,B,clk,reset,FinalSum,ci,co,px1,gx1,px2,gx2);
input clk,reset;
input [31:0] A,B;
output [31:0] FinalSum;
input         ci;
output        co;
output wire px1,gx1,px2,gx2;

wire carry_d1;
reg [15:0] Lsum_d1;
wire [15:0] Lsum_d1_nxt;
reg [15:0] Lsum_d2;
reg [15:0] Aup_d1,Bup_d2;
reg [15:0] Usum_d2;
wire [15:0] Usum_d2_nxt;
wire [31:0] FinalSum;
```

```
wire c16;
assign c16 = gx1 ^ (px1 && ci),
      co = gx2 ^ (px2 && c16);



CLA_16 CLA1(            //后十五位加法
                    .A(A[15:0]),
                    .B(B[15:0]),
                    .c0(ci),
                    .S(Lsum_d1_nxt),
                    .px(px1),
                    .gx(gx1)
                );



  CLA_16 CLA2(          //前 15 位加法，赋值在后面的 always 块中
                     .A(Aup_d1),
                     .B(Bup_d2),
                     .c0(carry_d1),
                     .S(Usum_d2_nxt),
                     .px(px2),
                     .gx(gx2)
                );



assign carry_d1 = c16;
assign FinalSum = {Usum_d2,Lsum_d2};

always@(posedge clk or negedge reset)begin//对中间变量进行前 15 位的赋值
 if(reset)begin
 Lsum_d1 <= 0;
```

```
 Lsum_d2 <= 0;
 Aup_d1 <= 0;
 Bup_d2 <= 0;
 Usum_d2 <= 0;
end
else begin
 Lsum_d1 <= Lsum_d1_nxt;
 Lsum_d2 <= Lsum_d1[15:0];
 Aup_d1 <= A[31:16];
 Bup_d2 <= B[31:16];
 Usum_d2 <= Usum_d2_nxt;
end
end
Endmodule
```

④16 位 CLA 模块（并行调用 4 个 4 位加法器）

```
module CLA_16(A,B,c0,S,px,gx);
   input [16:1] A;
                input [16:1] B;
                input c0;
                output gx,px;
                output [16:1] S;

                wire c4,c8,c12;
                wire Pm1,Gm1,Pm2,Gm2,Pm3,Gm3,Pm4,Gm4;

                adder_4 adder1(   //1~4 位加法
                    .x(A[4:1]),
                      .y(B[4:1]),
                      .c0(c0),
                      .c4(),
                      .F(S[4:1]),
                      .Gm(Gm1),
                      .Pm(Pm1)
                );
```

```
        adder_4 adder2(  //5~8 位加法
            .x(A[8:5]),
              .y(B[8:5]),
              .c0(c4),
              .c4(),
              .F(S[8:5]),
              .Gm(Gm2),
              .Pm(Pm2)
        );

  adder_4 adder3(     //9~12 位加法
              .x(A[12:9]),
              .y(B[12:9]),
              .c0(c8),
              .c4(),
              .F(S[12:9]),
              .Gm(Gm3),
              .Pm(Pm3)
        );

  adder_4 adder4(          //13~16 位加法
              .x(A[16:13]),
              .y(B[16:13]),
              .c0(c12),
              .c4(),
              .F(S[16:13]),
              .Gm(Gm4),
              .Pm(Pm4)
        );
        //进位的计算
        assign   c4 = Gm1 ^ (Pm1 & c0),
                 c8 = Gm2 ^ (Pm2 & Gm1) ^ (Pm2 & Pm1 & c0),
                c12 = Gm3 ^ (Pm3 & Gm2) ^ (Pm3 & Pm2 & Gm1) ^ (Pm3 & Pm2 &
Pm1 & c0);
```

```
  assign  px = Pm1 & Pm2 & Pm3 & Pm4,
  gx = Gm4 ^ (Pm4 & Gm3) ^ (Pm4 & Pm3 & Gm2) ^ (Pm4 & Pm3 & Pm2 & Gm1);

endmodule
```

④4 位加法器模块（并列调用 4 个 1 位加法器和 4 位 CLA 实现）

```
module adder_4(x,y,c0,c4,F,Gm,Pm);
    input [4:1] x;
                input [4:1] y;
                input c0;
                output c4,Gm,Pm;
                output [4:1] F;

                wire p1,p2,p3,p4,g1,g2,g3,g4;
                wire c1,c2,c3;
//各个位数同时相加，用四个 1 位加法器
                adder adder1(.X(x[1]), .Y(y[1]), .Cin(c0), .F(F[1]), .Cout());

                adder adder2(.X(x[2]), .Y(y[2]), .Cin(c1), .F(F[2]),.Cout());

                adder adder3(.X(x[3]),.Y(y[3]), .Cin(c2), .F(F[3]), .Cout());

                adder adder4( .X(x[4]), .Y(y[4]),.Cin(c3), .F(F[4]),.Cout());

                  CLA CLA(.c0(c0),.c1(c1),.c2(c2),.c3(c3),.c4(c4),
                          .p1(p1),.p2(p2),.p3(p3),.p4(p4),
                          .g1(g1),.g2(g2),.g3(g3),.g4(g4)
                  );



 assign   p1 = x[1] ^ y[1],p2 = x[2] ^ y[2],
        p3 = x[3] ^ y[3],p4 = x[4] ^ y[4];
```

```
  assign   g1 = x[1] & y[1],g2 = x[2] & y[2],
           g3 = x[3] & y[3], g4 = x[4] & y[4];

  assign Pm = p1 & p2 & p3 & p4,
      Gm = g4 ^ (p4 & g3) ^ (p4 & p3 & g2) ^ (p4 & p3 & p2 & g1);

endmodule
```

⑤1 位加法器模块和 4 位 CLA 模块

```
module adder(X,Y,Cin,F,Cout);//通过真值表列出输出进位与输入的逻辑公式即可实现 1 位加法器

  input X,Y,Cin;
  output F,Cout;

  assign F = X ^ Y ^ Cin;
  assign Cout = (X ^ Y) & Cin | X & Y;
endmodule

module CLA(c0,c1,c2,c3,c4,p1,p2,p3,p4,g1,g2,g3,g4);
  //通过列真值表列出进位 c1,c2,c3,c4 的计算
          input c0,g1,g2,g3,g4,p1,p2,p3,p4;
          output c1,c2,c3,c4;

          assign c1 = g1 ^ (p1 & c0),
             c2 = g2 ^ (p2 & g1) ^ (p2 & p1 & c0),
            c3 = g3 ^ (p3 & g2) ^ (p3 & p2 & g1) ^ (p3 & p2 & p1 & c0),
             c4=g4^(p4&g3)^(p4&p3&g2)^(p4&p3&p2&g1)^(p4&p3&p2&p1&c0);

endmodule
```

（2）仿真波形生成与观察

①仿真代码（由于顶层模块主要用于数码管的显示，仿真时实例化的模块是 j32-32 位加减法器模块，设置了不同的值用于

仿真）

```
module sim;


 reg [32:1] A;
 reg [32:1] B;
 reg c0;
 wire Gm;
 wire Pm;
 wire [32:1] S;
 wire c32;
 reg    clk;
 reg     rst_n;

 j32j d1( .in1(A),.in2(B),.add_sub(c0),  .Gm(Gm),.Pm(Pm),
             .sum(S),.c32(c32), .clk(clk),.rst_n(rst_n);


 initial begin
    clk   = 0;
    rst_n  = 1;
    @(posedge clk)   rst_n = 0;
        c0=0 ; A = 32'd1; B = 32'd1;
                @(posedge clk)  c0=0 ; A = 32'd25;  B = 32'd78;
                @(posedge clk)  c0=0 ; A = 32'd3432523;  B = 32'd3245325;
                @(posedge clk)  c0=0 ; A = 32'd11111;  B = 32'd11111;
                @(posedge clk)  c0=0 ; A = 32'd9999;  B = 32'd9999;
                @(posedge clk)  c0=0 ; A = 32'd456;  B = 32'd234;
                @(posedge clk);  c0=0 ; A = 32'd245;  B = 32'd678;


                @(posedge clk)   c0=0 ; A = 32'd0; B = 32'd0;
                @(posedge clk);  c0=0 ; A = 32'd25;  B = 32'd78;
                @(posedge clk)  c0=0 ; A = 32'd3423;  B = 32'd3245325;
                @(posedge clk);  c0=0 ; A = 32'd11111;  B = 32'd11111;
                @(posedge clk)  c0=0 ; A = 32'd99;  B = 32'd9999;
                @(posedge clk)  c0=0 ; A = 32'd45;  B = 32'd234;
```

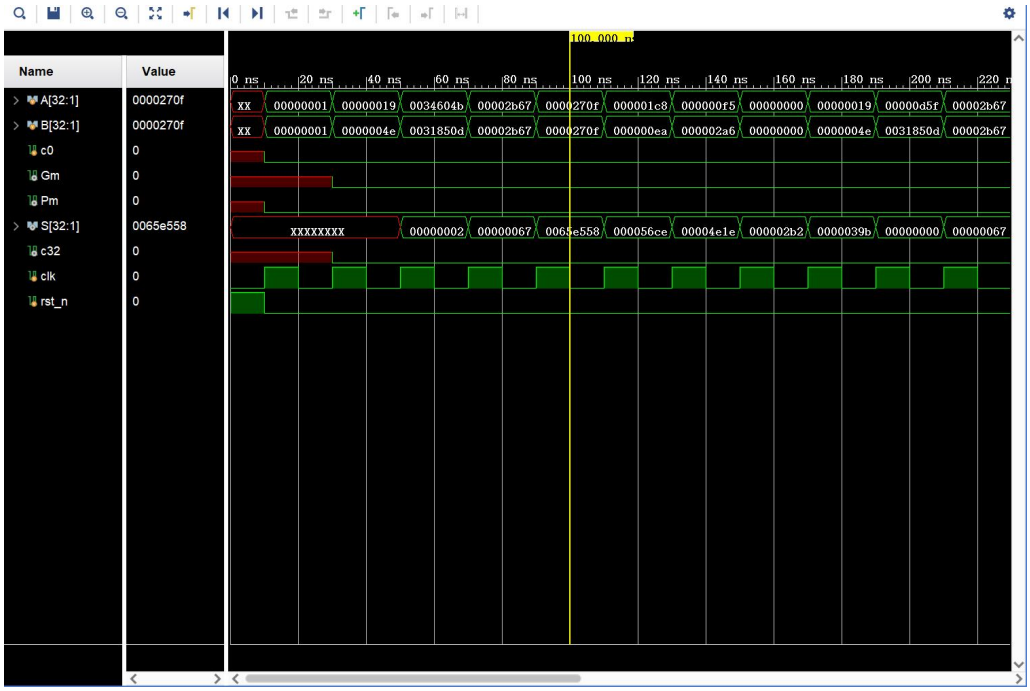
```
                @(posedge clk)  c0=0 ; A = 32'd245;  B = 32'd678;
                repeat(10) @(posedge clk);
    $finish;
end


always #10  clk = ~clk;


endmodule
```

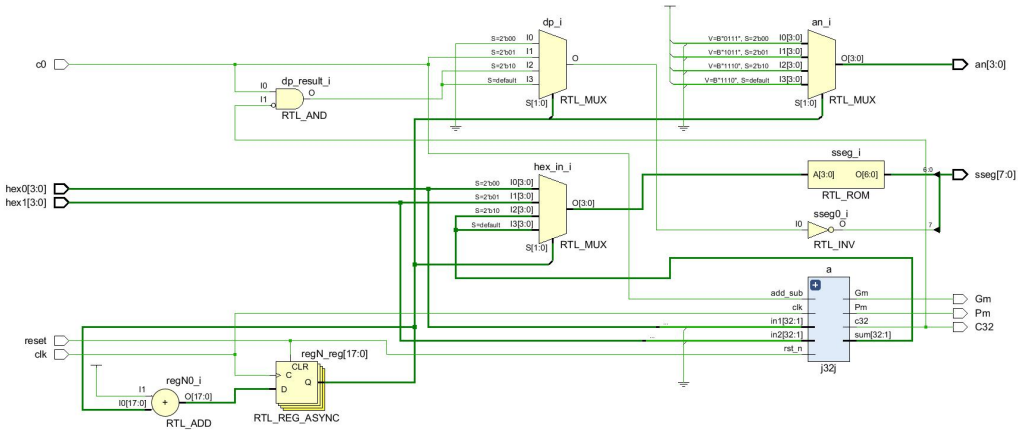
②仿真波形

运行行为仿真，仿真波形如下图，符合设计预期。由于该加法器的结构是二级流水线并行加法器，在一个周期内不能将两次结果计算出来再相加，所以乘法器的输出有两个周期的延迟。



（3）RTL 分析

利用 vivado 自带的 RTL 分析功能，对项目代码综合后进行 RTL 电路分析，如下图所示。



RTL 分析电路图

（4）综合、绑定引脚、实现

约束文件如下：

```
set_property IOSTANDARD LVCMOS33 [get_ports c0]
set_property IOSTANDARD LVCMOS33 [get_ports C32]
set_property IOSTANDARD LVCMOS33 [get_ports Gm]
set_property IOSTANDARD LVCMOS33 [get_ports Pm]
set_property PACKAGE_PIN R2 [get_ports {c0}]
set_property PACKAGE_PIN P1 [get_ports {C32}]
set_property PACKAGE_PIN P3 [get_ports {Gm}]
set_property PACKAGE_PIN N3 [get_ports {Pm}]


set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex0[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex0[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex0[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex0[0]}]
```

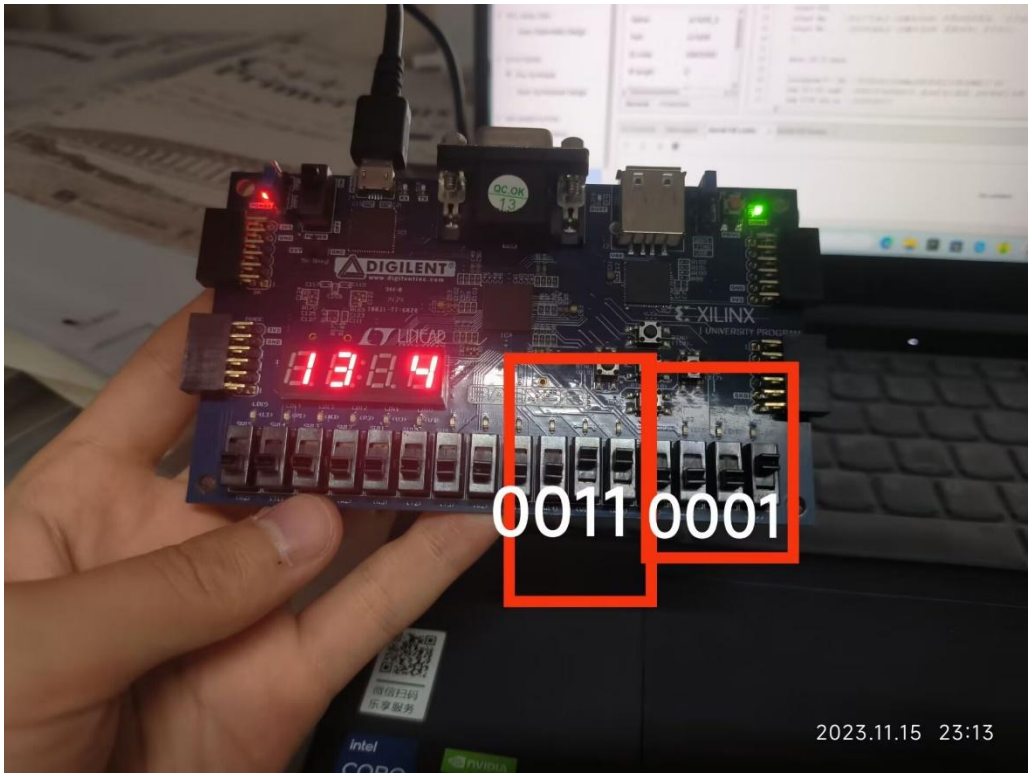
```
set_property IOSTANDARD LVCMOS33 [get_ports {hex1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {hex1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sseg[0]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
set_property PACKAGE_PIN U2 [get_ports {an[0]}]
set_property PACKAGE_PIN V17 [get_ports {hex0[0]}]
set_property PACKAGE_PIN V16 [get_ports {hex0[1]}]
set_property PACKAGE_PIN W16 [get_ports {hex0[2]}]
set_property PACKAGE_PIN W17 [get_ports {hex0[3]}]
set_property PACKAGE_PIN W15 [get_ports {hex1[0]}]
set_property PACKAGE_PIN V15 [get_ports {hex1[1]}]
set_property PACKAGE_PIN W14 [get_ports {hex1[2]}]
set_property PACKAGE_PIN W13 [get_ports {hex1[3]}]
set_property PACKAGE_PIN W7 [get_ports {sseg[6]}]
set_property PACKAGE_PIN W6 [get_ports {sseg[5]}]
set_property PACKAGE_PIN U8 [get_ports {sseg[4]}]
set_property PACKAGE_PIN V8 [get_ports {sseg[3]}]
set_property PACKAGE_PIN U5 [get_ports {sseg[2]}]
set_property PACKAGE_PIN V5 [get_ports {sseg[1]}]
set_property PACKAGE_PIN U7 [get_ports {sseg[0]}]


set_property PACKAGE_PIN V7 [get_ports {sseg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN U1 [get_ports reset]


set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk]
```

（5）生成比特流，打开硬件程序并打开硬件目标，在开发板上验证自己的逻辑程序。



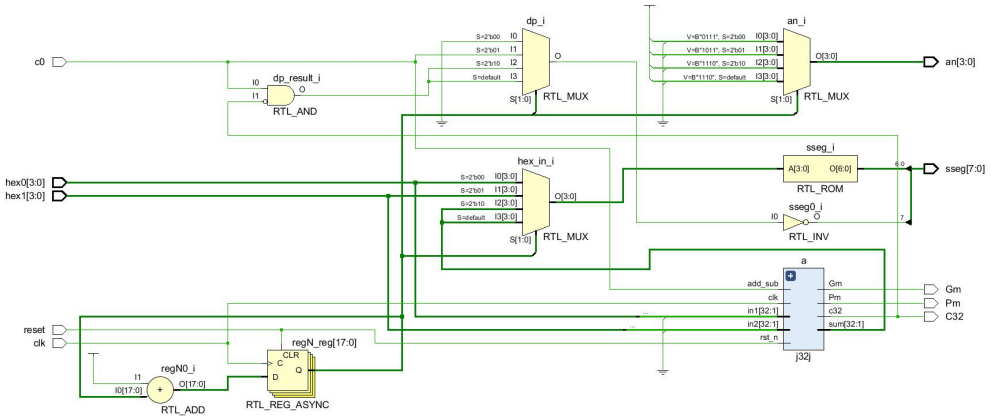
通过实验测试，当 hex0=2'b0001,hex1=2;b0011 时，结果计算为 4，显示的灯结果显示正确，证明所做实验正确。

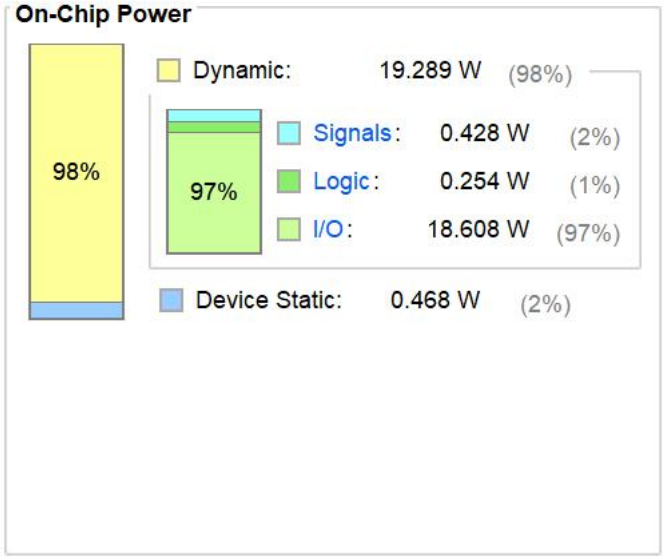
五、实验过程中遇到的问题及解决情况(主要问题及解决情况)

在刚开始的时候我们编写 32 位加法器的时候，没有想到使用多级加法器叠加出 32 级加法器，而是想要直接用 for 语句设计出加法器，这倒是我们代码的冗长并且极易容易出错。之后我们在绑定引脚的时候三个灯绑定重复了，导致后期上版验证的时候结果出错，后来我们调整了引脚的绑定。

六、实验结果及分析和（或）源程序调试过程

1.本次实验的电路和功耗分析

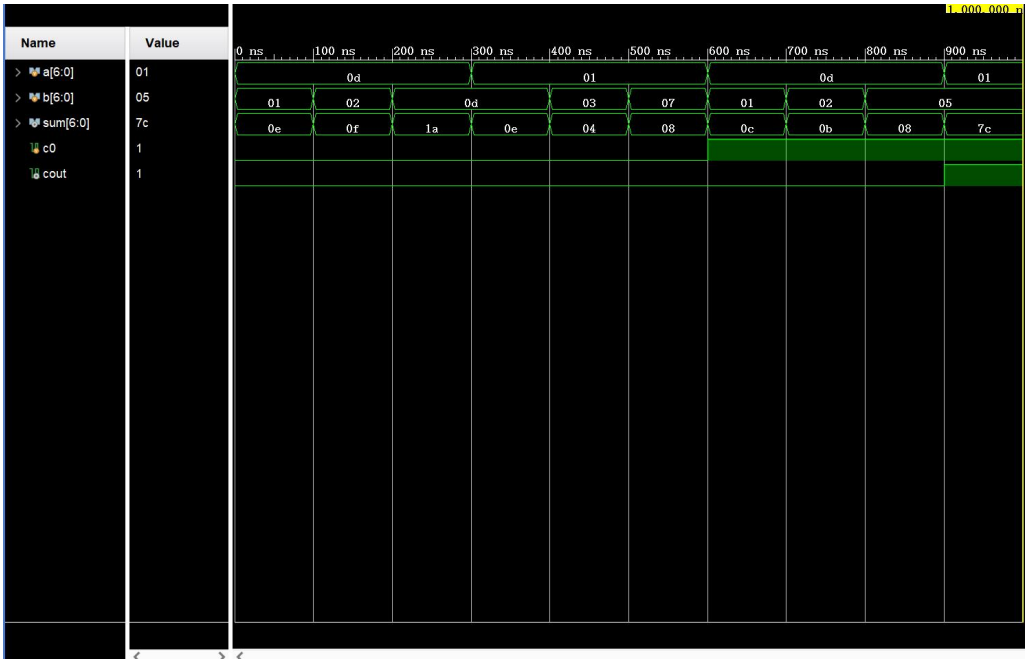




由于本次实验是 32 位的加减法器，输入输出端口较多，I/O 资源占比较大。同时本次实验中采用了流水线的结构，所以 logic 资源和 signal 资源消耗比一般的项目少。

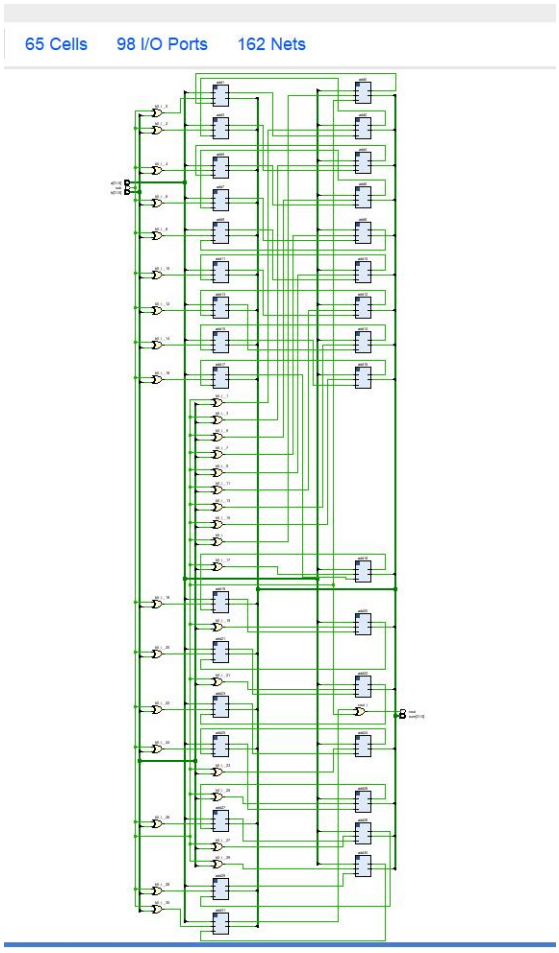
2.与实验六：加减法器的设计中的 32 位加减法器对比

（1）从逻辑结构上，实验六中的加减法器是串行的，只有上一个加法的结果出来之后才能进行下一项的加法；而此次实验中的加减法器的流水线结构是并行的，可以同时计算加减法结果然后进行计算，效率更高，功耗更小。



（3）电路分析对比

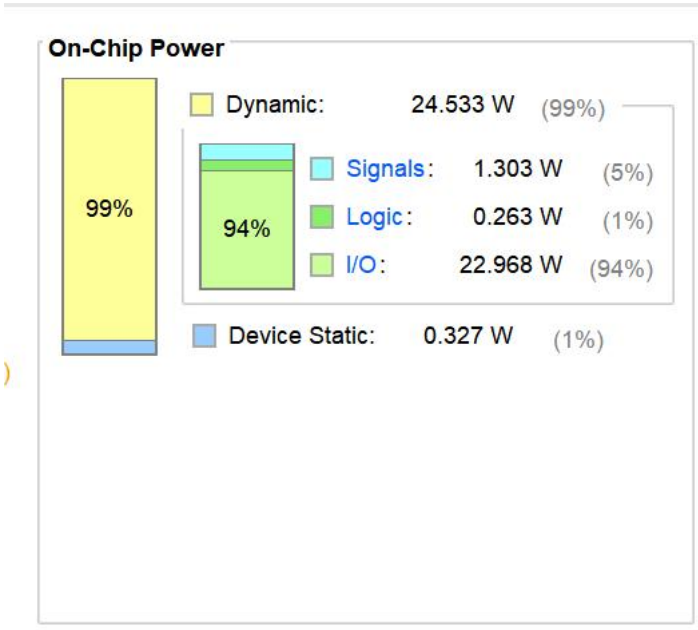
实验六中的加减法器 RTL 电路分析结果如下



由于该电路直接由 32 个加法器串行组成，所以生成的电路相对复杂

（4）功耗分析对比

实验六中的加减法器 RTL 电路分析结果如下



可以看到，流水加法器的两级流水线结构使得 signals 资源消耗显著减少。

七、小组分工情况说明

：编写 32 位流水线加法器设计的顶层代码，并进行仿真验证。共同分析在 RTL 分析、逻辑 资源占用、功耗上的问题。

：编写 32 位流水线加法器的 4 位加法器和 CLA 代码，并进行仿真验证。共同分析在 RTL 分析、逻辑 资源占用、功耗上的问题。

：编写实验六的组合逻辑加减法器代码，编写系统加法器的代码，完成引脚 绑定和上板验证。共同分析在 RTL 分析、逻辑 资源占用、功耗上的问题。