****

**计算机组成原理 ·实验报告·**

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机组成原理 |
| 姓 名： | 党存远 |
| 学 院： | 计算机科学与技术 |
| 专 业： | 物联网工程 |
| 学 号： | 2022217587 |
| 指导教师： | 丁贤庆 |

2024 年 6月 8 日

**合肥工业大学实验报告**

学生姓名： 党存远 班级： 物联网工程22-2 学号： 2022217587

实验地点： 电气实验楼507 实验日期： 2024 年 6 月 8 日

**实验五 在vivado中进行运算器的设计**

一、实验目的及环境

1、掌握一位全加器的工作原理和逻辑功能

2、掌握串行进位加法器的工作原理和进位延迟

3、掌握减法器的实现原理

4、掌握加减法器的设计方法

5、掌握ALU基本原理及在CPU中的作用

6、掌握ALU的设计方法

7、装有vivado的计算机 1台

8、EGO1开发板 1块

二、实验目标及任务

1、采用原理图方式设计4位加/减法器

2、实现4位ALU及应用设计

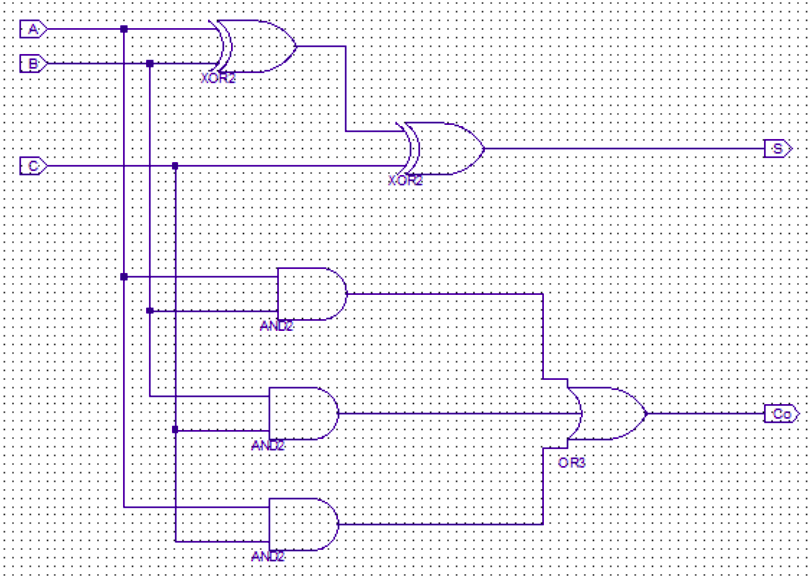
三、实验过程及记录



本节重点介绍实验的具体过程，包括：代码设计层次结构图及说明、源代码（包括注释）、PC机上进行的关键步骤截图及说明、调试过程等，这部分的内容应当与实际操作过程类似即可（简单明了）。

1、根据实验指导书中的要求首先需要采用4个一位的全加器和附加逻辑设计两个四位数的加/减法器因此需要依次设计四位加/减法器

1位全加器的设计

①电路原理图:

由上述电路原理图可得，我们需要使用3个与门、2个异或门和一个或门来实现1位全加器，接下来按照电路原理图，使用代码将信号线连接即可。

②源代码:

`timescale 1ns / 1ps

module adder\_1bit(

    input a,

    input b,

    input ci,

    output s,

    output co

    );

    and m0(c1, a, b);

    and m1(c2, b, ci);

    and m2(c3, a, ci);

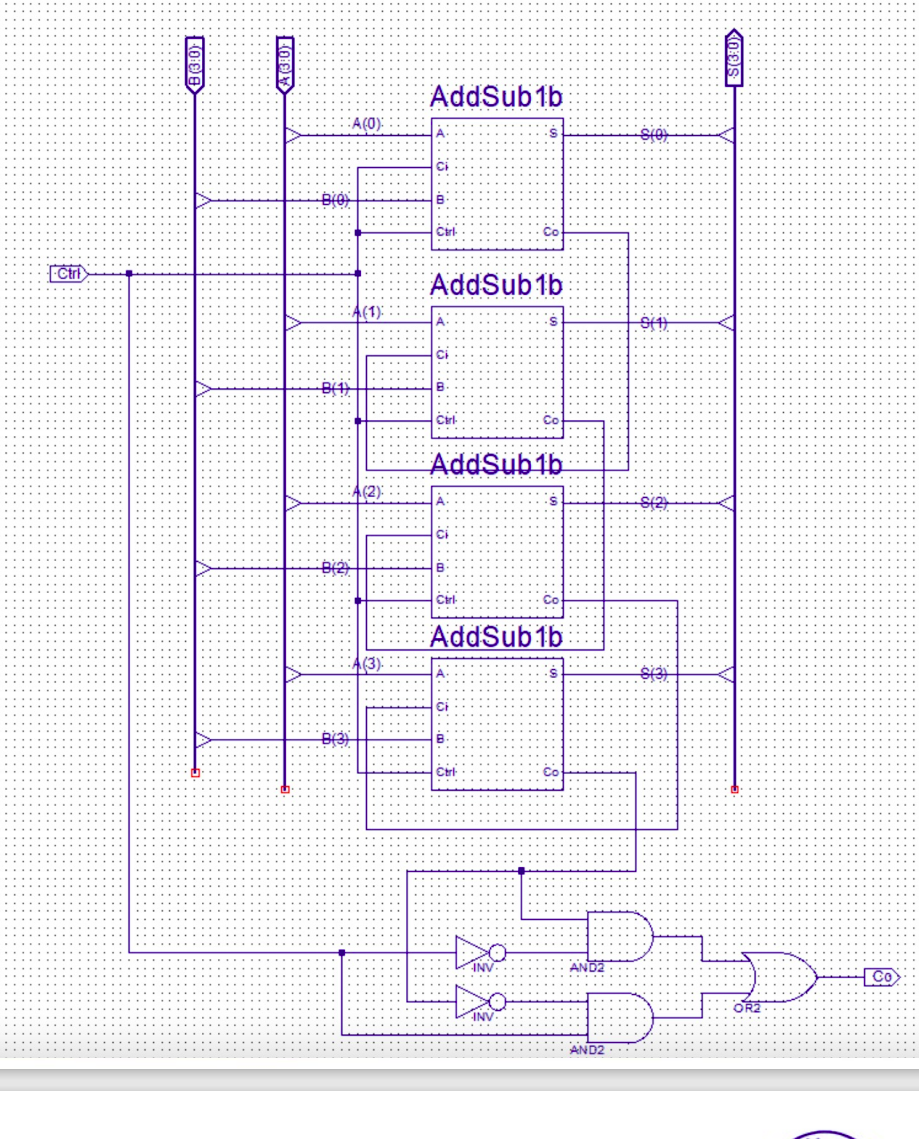
    xor m3(s1, a, b);

    xor m4(s, s1, ci);

    or m5(co, c1, c2, c3);

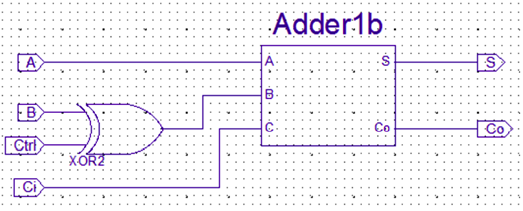
endmodule

采用四个一位全加器和附加逻辑设计两个四位数的加/减法器

①电路原理图:

根据电路原理图可得，我们需要使用四个一位加减法器串联来实现一个四位的加减法器，因此，我们先根据一位加减法的电路原理图来实现一位加法减器，再根据电路图，例化四个一位加法器，并将其串联，即可得到一个四位加减法器。

1位加/减法器电路:



和上述相同，结合之前实现的一位加法器，实现一位加减法器。

②源代码:

一位加减法器:

`timescale 1ns / 1ps

module addsub\_1bit(

    input a,

    input b,

    input ctrl,

    input ci,

    output s,

    output co

    );

    xor x(xor0, b, ctrl);

    adder\_1bit adder(.a(a), .b(xor0), .ci(ci), .s(s), .co(co));

endmodule

四位加减法器:

`timescale 1ns / 1ps

module addsub\_4bit(

    input [3:0] a, b,

    input ctrl,

    output [3:0] s,

    output co

    );

    // 例化四个一位加减法器，并进行串联

    addsub\_1bit as0(.a(a[0]),.b(b[0]),.ci(ctrl),.ctrl(ctrl),.s(s[0]),.co(co0));

    addsub\_1bit as1(.a(a[1]),.b(b[1]),.ci(co0),.ctrl(ctrl),.s(s[1]),.co(co1));

    addsub\_1bit as2(.a(a[2]),.b(b[2]),.ci(co1),.ctrl(ctrl),.s(s[2]),.co(co2));

    addsub\_1bit as3(.a(a[3]),.b(b[3]),.ci(co2),.ctrl(ctrl),.s(s[3]),.co(co3));

    // 输出进位

    not not0(nc,ctrl);

    not not1(nco, co3);

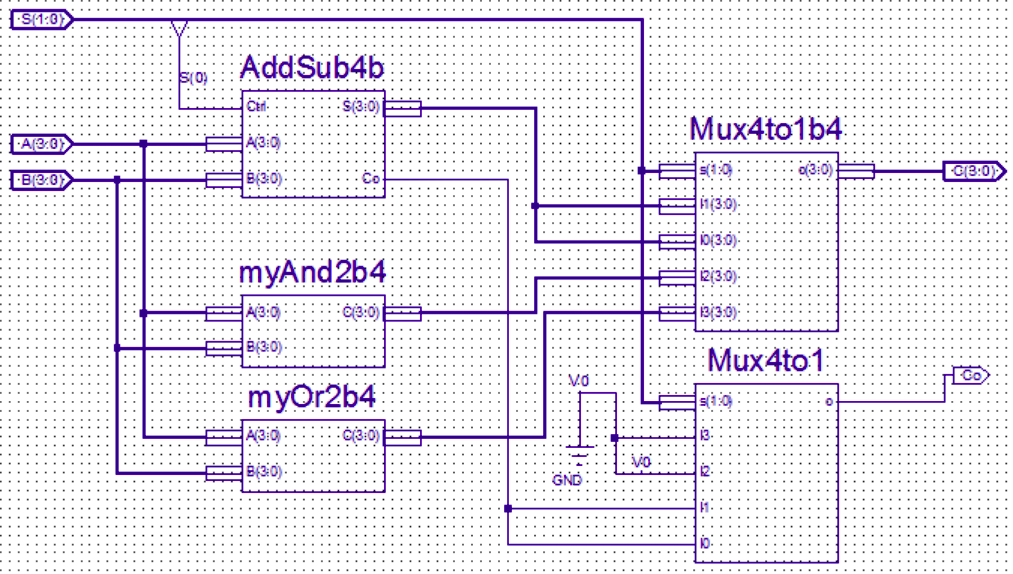
    and and0(a0,co3,nc);

    and and1(a1, nco, ctrl);

    or or0(co, a0,a1);

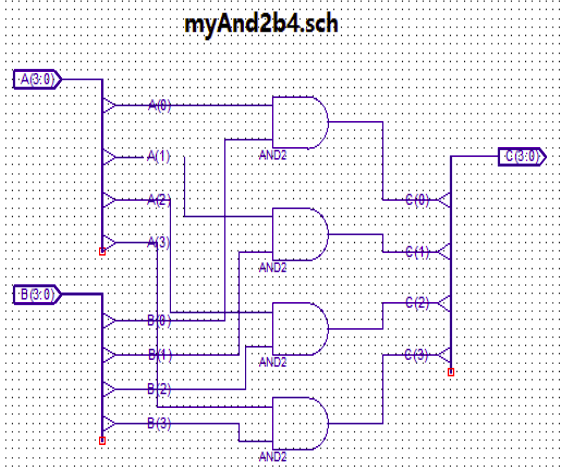
endmodule

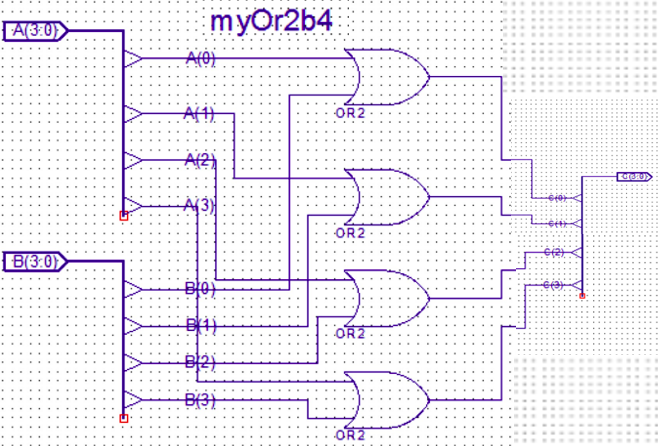
采用与逻辑、或逻辑和数据选择器设计两个四位数的加/减法器

①电路原理图:

根据电路原理图，我们需要首先实现四位的与逻辑、或逻辑门和1位、4位的数据选择器。

电路原理图:





②源代码:

4位与逻辑门:

`timescale 1ns / 1ps

module and\_4bit(

    input [3:0] a, b,

    output reg[3:0] c

    );

    always @\* begin

        c = a & b;

    end

endmodule

4位或逻辑门:

`timescale 1ns / 1ps

module or\_4bit(

    input [3:0] a, b,

    output reg[3:0] c

    );

    always @\* begin

        c = a | b;

    end

endmodule

1位4选1选择器:

`timescale 1ns / 1ps

module mux41\_1bit(

  input i0, i1, i2, i3,

  input [1:0] s,

  output reg o

);

  always @\* begin

    case(s)

      2'b00: o = i0;

      2'b01: o = i1;

      2'b10: o = i2;

      2'b11: o = i3;

    endcase

  end

endmodule

4位4选1选择器:

`timescale 1ns / 1ps

module mux41\_4bit(

    input [3:0] i0, i1, i2, i3,

    input [1:0] s,

    output reg [3:0] o

    );

    always @\* begin

        case(s)

            2'b00: o = i0;

            2'b01: o = i1;

            2'b10: o = i2;

            2'b11: o = i3;

        endcase

    end

endmodule

4位加/减法器:

`timescale 1ns / 1ps

module addsub\_4bitmux(

    input [1:0] s,

    input [3:0] a, b,

    output [3:0]o,

    output co

    );

    // 声明数据线

    wire[3:0] st, cot, c0, c1;

    // 例化元件，并连接数据线

    addsub\_4bit addsub4b(.a(a),.b(b),.ctrl(s[0]),.s(st),.co(cot));

    and\_4bit and4b(.a(a),.b(b),.c(c0));

    or\_4bit or4b(.a(a),.b(b),.c(c1));

    mux41\_4bit mux41b4(.i0(st),.i1(st),.i2(c0),.i3(c1),.s(s),.o(o));

    mux41\_1bit mux41(.i0(cot),.i1(cot),.i2(1'b0),.i3(1'b0),.s(s),.o(co));

endmodule

四、实验结果分析

1.这里应给出相应的实验结果。分析应有条理，要求采用规范的书面语。

2.每个实验都需要做模拟仿真，需要对仿真波形进行简单的文字说明。

3.对下载到开发板上的图片结果做分析说明。

4.原则上要求使用图片与文字结合的形式说明，因为word和PDF文档不支持视频，所以请不要使用视频文件。

5.图片请在垂直方向，不要横向。不要用很大的图片（不要超过1M），请先做裁剪操作。

1、1位全加器的设计:

①激励文件:

`timescale 1ns / 1ps

module adder\_tb();

    reg a;

    reg b;

    reg ci;

    wire s;

    wire co;

    adder\_1bit u0(.a(a), .b(b), .ci(ci), .s(s), .co(co));

    initial

    begin

        a = 1'b0;

        b = 1'b0;

        ci = 1'b0;

        #100;

        a = 1'b0;

        b = 1'b0;

        ci = 1'b1;

        #100;

        a = 1'b0;

        b = 1'b1;

        ci = 1'b0;

        #100;

        a = 1'b0;

        b = 1'b1;

        ci = 1'b1;

        #100

        a = 1'b1;

        b = 1'b0;

        ci = 1'b0;

        #100;

        a = 1'b1;

        b = 1'b0;

        ci = 1'b1;

        #100;

        a = 1'b1;

        b = 1'b1;

        ci = 1'b0;

        #100;

        a = 1'b1;

        b = 1'b1;

        ci = 1'b1;

        #100;

        a = 1'b0;

        b = 1'b0;

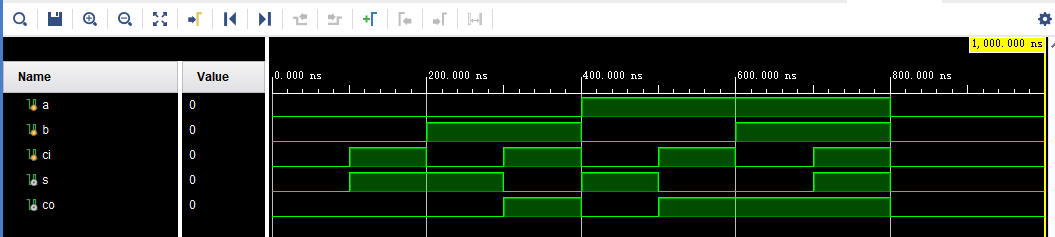
        ci = 1'b0;

        #100;

    end

endmodule

②仿真波形:



③仿真结果解释:

如图所示，设计1位全加器，0+0+0=0+0；0+0+1=1+0；0+1+0=1+0；0+1+1=0+1；1+0+0=1+0；1+0+1=0+1；1+1+1=1+1；

④约束文件的编写:

set\_property PACKAGE\_PIN R1 [get\_ports {a}]

set\_property PACKAGE\_PIN N4 [get\_ports {b}]

set\_property PACKAGE\_PIN M4 [get\_ports {ci}]

set\_property PACKAGE\_PIN G4 [get\_ports {s}]

set\_property PACKAGE\_PIN F6 [get\_ports {co}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a}]

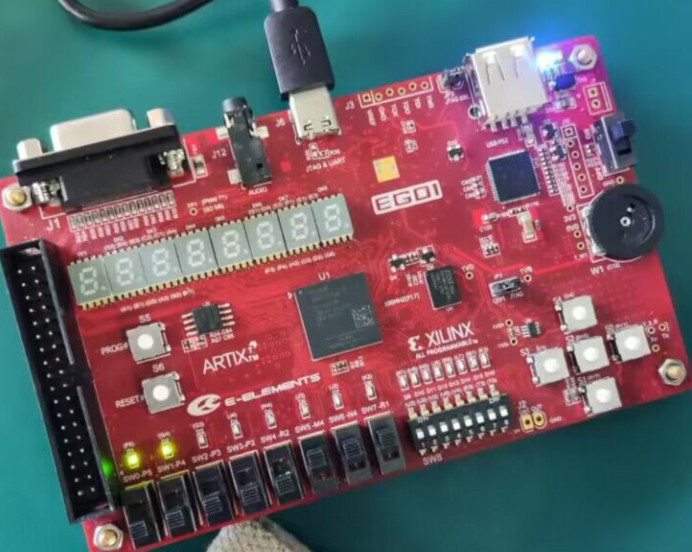
set\_property IOSTANDARD LVCMOS33 [get\_ports {b}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ci}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {co}]

⑤开发板运行图片:



2、采用四个一位的全加器和附加逻辑设计两个四位数的加/减法器

①激励文件:

`timescale 1ns / 1ps

module addsub\_4bit\_tb();

    // Instantiate the DUT

    // Define inputs

    reg [3:0] A;

    reg [3:0] B;

    reg Ctrl;

    // Define outputs

    wire [3:0] s;

    wire co;

    addsub\_4bit dut(

        .a(A),

        .b(B),

        .ctrl(Ctrl),

        .s(s),

        .co(co)

    );

    // Initialize inputs

    initial begin

        A = 4'b0000;

        B = 4'b0000;

        Ctrl = 0;

        end

    always begin

        #10 A = 4'b0001; B = 4'b0010; Ctrl = 0;

        // Subtract test case

        #10 A = 4'b0011; B = 4'b0001; Ctrl = 1;

        // Test case with carry

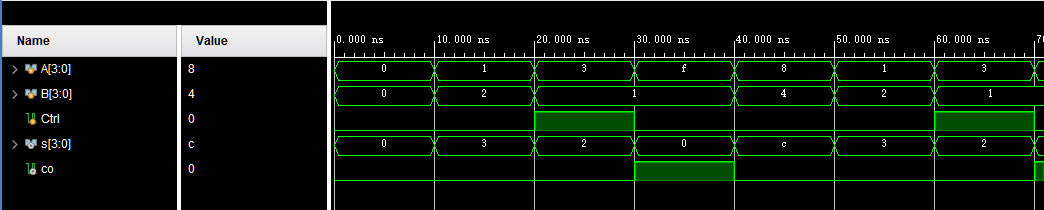
        #10 A = 4'b1111; B = 4'b0001; Ctrl = 0;

        // Test case with borrow

        #10 A = 4'b1000; B = 4'b0100; Ctrl = 0;

    end

②仿真波形:



③仿真结果解释: 0+0=0+0；1+2=3+0；3-1=2；f+1=0+1（进位）；8+4=c+0；

④约束文件:

set\_property PACKAGE\_PIN P5 [get\_ports {a[3]}]

set\_property PACKAGE\_PIN P4 [get\_ports {a[2]}]

set\_property PACKAGE\_PIN P3 [get\_ports {a[1]}]

set\_property PACKAGE\_PIN P2 [get\_ports {a[0]}]

set\_property PACKAGE\_PIN R2 [get\_ports {b[3]}]

set\_property PACKAGE\_PIN M4 [get\_ports {b[2]}]

set\_property PACKAGE\_PIN N4 [get\_ports {b[1]}]

set\_property PACKAGE\_PIN R1 [get\_ports {b[0]}]

set\_property PACKAGE\_PIN U3 [get\_ports {ctrl}]

set\_property PACKAGE\_PIN F6 [get\_ports {s[3]}]

set\_property PACKAGE\_PIN G4 [get\_ports {s[2]}]

set\_property PACKAGE\_PIN G3 [get\_ports {s[1]}]

set\_property PACKAGE\_PIN J4 [get\_ports {s[0]}]

set\_property PACKAGE\_PIN H4 [get\_ports {co}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {ctrl}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {co}]

⑤开发板运行图片:



3、采用与逻辑、或逻辑和数据选择器设计两个四位数的加/减法器

①激励文件:

`timescale 1ns / 1ps

module addsub\_4bitmux\_tb();

    reg [1:0] s;

    reg [3:0] a, b;

    wire [3:0] o;

    wire co;

    addsub\_4bitmux addsub4bmux(.s(s), .a(a), .b(b), .o(o), .co(co));

    initial begin

        #0

        s = 2'b00;

        a = 4'b0000;

        b = 4'b0000;

        #10

        s = 2'b00;

        a = 9;

        b = 4'b0110;

        #10

        s = 2'b01;

        a = 4'b1010;

        b = 4'b0110;

        #10

        s = 2'b10;

        a = 4'b0101;

        b = 4'b0011;

        #10

        s = 2'b11;

        a = 4'b0101;

        b = 4'b0011;

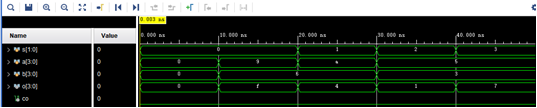
        #10

        $finish;

    end

endmodule

②仿真波形:



③仿真结果解释: 0+0=0；9+6=f；a-6=4；5-3-1=1；5-1+3=7；

④约束文件:

set\_property PACKAGE\_PIN P5 [get\_ports {a[3]}]

set\_property PACKAGE\_PIN P4 [get\_ports {a[2]}]

set\_property PACKAGE\_PIN P3 [get\_ports {a[1]}]

set\_property PACKAGE\_PIN P2 [get\_ports {a[0]}]

set\_property PACKAGE\_PIN R2 [get\_ports {b[3]}]

set\_property PACKAGE\_PIN M4 [get\_ports {b[2]}]

set\_property PACKAGE\_PIN N4 [get\_ports {b[1]}]

set\_property PACKAGE\_PIN R1 [get\_ports {b[0]}]

set\_property PACKAGE\_PIN F6 [get\_ports {o[3]}]

set\_property PACKAGE\_PIN G4 [get\_ports {o[2]}]

set\_property PACKAGE\_PIN G3 [get\_ports {o[1]}]

set\_property PACKAGE\_PIN J4 [get\_ports {o[0]}]

set\_property PACKAGE\_PIN U3 [get\_ports {s[1]}]

set\_property PACKAGE\_PIN U2 [get\_ports {s[0]}]

set\_property PACKAGE\_PIN H4 [get\_ports {co}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {a[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {b[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {o[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {o[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {o[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {o[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {s[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {co}]

⑤开发板运行图片:



五、遇到的问题与实验心得体会

简要地叙述一下实验过程中的感受，以及其他的问题描述和自己的感想。特别是实验中遇到的困难，最后如何解决的。在用verilog代码写程序时遇到语法或其他错误，如何修改解决的。

在本次的计算机组成实验中，我进行了一系列的工作，包括根据4位加/减法器的原理图，实现了4位ALU以及在FPGA开发板上运行代码。这个实验过程中，我积累了很多宝贵的经验，并且对计算机组成原理有了更深入的理解，了解了ALU是如何工作的。

在本次实验中，遇到的困难主要在于约束文件的编写，仅在参考指导书的过程中，直接的从参考书上照搬代码，却漏写了空格或引脚书写错误，导致后续的生成比特流阶段报错，却难以找到错误。

另外，对于verilog语言的不熟悉，导致在编写的过程中，出现了很多语法上的错误，在编写的过程中，还需要不断的翻阅文档，了解如何对元件进行例化，还有一些变量声明上的注意事项。经过这次实验后，对于verilog的使用也更加熟练了。

vivado这个软件本身就具有复杂性，在一开始，还需要不断借助指导书，才能创建一个正确的项目，在后续熟练后，也可以很快的创建项目，并向其中添加自己的代码了。

经过本次实验，我认为最重要的是要看懂电路原理图，理解了电路原理图，我们就能很快的实现出电路原理图中的每一个模块，并且最后将每一个模块相联，完成最终的功能实现。

总的来说，这个计算机组成实验给我带来了很多挑战，但也让我获得了很多宝贵的经验和技能。通过不断尝试和修正，我逐渐掌握了设计和实现加/减法器、ALU以及在FPGA上运行代码的方法。我深刻认识到了实践的重要性，并且学会了如何在面对困难时保持耐心和解决问题的能力。这个实验让我更加熟悉计算机组成原理，并对我今后的学习和工作具有很大的帮助。

**实验六 8位乘法器设计与实现**

一、实验目的及环境

1、了解16位有符号、无符号乘法器的实现原理。

2、使用Verilog实现16位无符号乘法器和有符号乘法器。

3、装有vivado的计算机 1台

4、EGO1开发板 1块

二、实验目标及任务

1、设计无符号乘法器，将两个8位无符号数相乘,得到一个16位无符号数。

2、设计有符号乘法器，将两个8位有符号数相乘,得到一个16位有符号数。

3、设计按键输入转换电路，设计七段显示器显示和控制电路： 通过8个按键输入两个4位的二进制乘数A和B，两个乘数和乘积的结果，分别送给多个七段显示器进行显示。

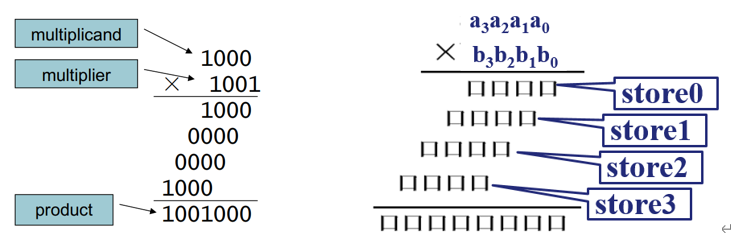
三、实验过程及记录



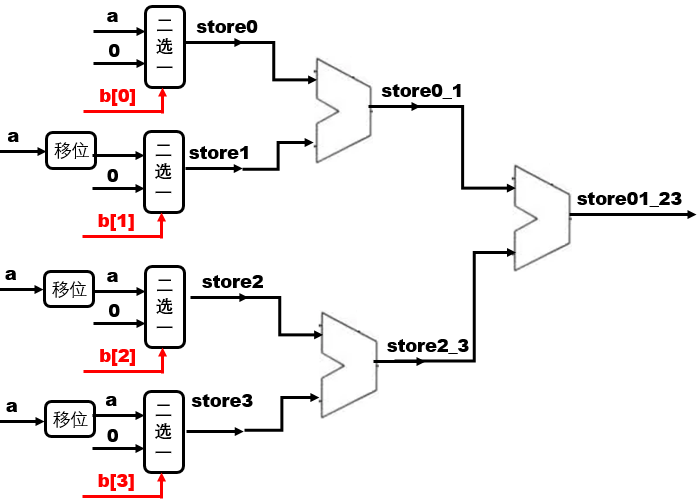
本节重点介绍实验的具体过程，包括：代码设计层次结构图及说明、源代码（包括注释）、PC机上进行的关键步骤截图及说明、调试过程等，这部分的内容应当与实际操作过程类似即可（简单明了）。

下图的乘法过程是手算的一个步骤，被乘数x是1000，乘数y为1001，在使用计算机实现乘法运算时，我们就可以通过模拟这个手算的过程来实现我们的乘法运算操作。

①电路原理图



根据该手工计算过程，得到电路原理图:



根据该电路原理图，我们可以相似的得到两个16位无符号数相乘的电路，根据该电路，我们即可通过代码来实现该乘法器。

②源代码:

16位乘法器:

`timescale 1ns / 1ps

module MULTU(

    input clk,    //乘法器时钟信号

    input reset,

    input [ 15:0]  a,      //输入a(被乘数)

    input [15:0]  b,      //输入b(乘数)

    output [ 31:0]  z     //乘积输出z

    );

    reg [31:0] temp;    //申请寄存器

    reg [31:0] stored [15:0];

    reg [31:0] addi\_i1 [7:0];

    reg [31:0] additi1\_i2ti3[3:0];

    reg [31:0] add[0:1];

    integer i;

    always @(posedge clk or negedge reset) begin

        // 初始化

        if(reset) begin

            temp <= 0;

            for (i = 0;i < 16;i = i + 1) begin

                stored[i] = 0;

            end

            for (i = 0;i < 8;i = i + 1) begin

                addi\_i1[i] = 0;

            end

            for (i = 0;i < 4;i = i + 1) begin

                additi1\_i2ti3[i] = 0;

            end

        end

        // 计算过程

        else begin

            stored[0] = b[0] ? {16'b0, a} : 32'b0;

            stored[1] = b[1] ? {15'b0, a, 1'b0} : 32'b0;

            stored[2] = b[2] ? {14'b0, a, 2'b0} : 32'b0;

            stored[3] = b[3] ? {13'b0, a, 3'b0} : 32'b0;

            stored[4] = b[4] ? {12'b0, a, 4'b0} : 32'b0;

            stored[5] = b[5] ? {11'b0, a, 5'b0} : 32'b0;

            stored[6] = b[6] ? {10'b0, a, 6'b0} : 32'b0;

            stored[7] = b[7] ? {9'b0, a, 7'b0} : 32'b0;

            stored[8] = b[8] ? {8'b0, a, 8'b0} : 32'b0;

            stored[9] = b[9] ? {7'b0, a, 9'b0} : 32'b0;

            stored[10] = b[10] ? {6'b0, a, 10'b0} : 32'b0;

            stored[11] = b[11] ? {5'b0, a, 11'b0} : 32'b0;

            stored[12] = b[12] ? {4'b0, a, 12'b0} : 32'b0;

            stored[13] = b[13] ? {3'b0, a, 13'b0} : 32'b0;

            stored[14] = b[14] ? {2'b0, a, 14'b0} : 32'b0;

            stored[15] = b[15] ? {1'b0, a, 15'b0} : 32'b0;

            for(i = 0;i < 8;i = i + 1)begin

                addi\_i1[i] = stored[2 \* i] + stored[2 \* i + 1];

            end

            for(i = 0;i < 4;i = i + 1) begin

                additi1\_i2ti3[i] = addi\_i1[2 \* i] + addi\_i1[2 \* i + 1];

            end

            for(i = 0;i < 2;i = i + 1) begin

                add[i] = additi1\_i2ti3[2 \* i] + additi1\_i2ti3[2 \* i + 1];

            end

            temp = add[0] + add[1];

        end

    end

    assign z = temp;

endmodule

时钟模块:

`timescale 1ns / 1ps

*//数码管动态扫描所需要的时钟模块*

module CLK\_DIV (

    input  clk0,

    output clk

);

    reg clk = 0;

    parameter N = 32'd25000;

    parameter WIDTH = 32 - 1;

    reg [WIDTH : 0] number = 0;

    always @(posedge clk0) begin

        if (number == N - 1) begin

            number <= 0;

            clk <= ~clk;

        end else begin

            number <= number + 1;

        end

    end

endmodule

4位8421码转换:

`timescale 1ns / 1ps

module Bit4\_to\_8421 (

    input      [3:0] a,

    output reg [7:0] bcd

);

    reg [3:0] temp1, temp2;

    always @(\*) begin

        temp1 <= a / 10;

        temp2 <= a - temp1 \* 10;

        bcd   <= {temp1, temp2};

    end

endmodule

8位8421码转换:

`timescale 1ns / 1ps

module Bit8\_to\_8421 (

    input      [ 7:0] a,

    output reg [15:0] bcd

);

    reg [3:0] temp1, temp2, temp3, temp4;

    always @(\*) begin

        temp1 <= a / 1000;

        temp2 <= (a - temp1 \* 1000) / 100;

        temp3 <= (a - temp1 \* 1000 - temp2 \* 100) / 10;

        temp4 <= a - temp1 \* 1000 - temp2 \* 100 - temp3 \* 10;

        bcd   <= {temp1, temp2, temp3, temp4};

    end

endmodule

数码管显示模块:

`timescale 1ns / 1ps

*//控制8个数码管动态显示的模块。segn[3:0]是8421BCD码。*

module ShowEightSeg7 (

    input            clk,

    input      [3:0] seg0,

    input      [3:0] seg1,

    input      [3:0] seg2,

    input      [3:0] seg3,

    input      [3:0] seg4,

    input      [3:0] seg5,

    input      [3:0] seg6,

    input      [3:0] seg7,

    output reg [7:0] an,

    output reg [7:0] segout\_1,

    output reg [7:0] segout\_2

);

    reg [2:0] state = 0;

    reg [3:0] bcd = 0;

    always @(posedge clk) begin

        state <= state + 1;

    end

    always @(\*) begin

        case (state)

            0: begin

                an  = 8'b00000001;

                bcd = seg0;

            end

            1: begin

                an  = 8'b00000010;

                bcd = seg1;

            end

            2: begin

                an  = 8'b00000100;

                bcd = seg2;

            end

            3: begin

                an  = 8'b00001000;

                bcd = seg3;

            end

            4: begin

                an  = 8'b00010000;

                bcd = seg4;

            end

            5: begin

                an  = 8'b00100000;

                bcd = seg5;

            end

            6: begin

                an  = 8'b01000000;

                bcd = seg6;

            end

            7: begin

                an  = 8'b10000000;

                bcd = seg7;

            end

            default: begin

                an  = 8'b0000000;

                bcd = 4'b0000;

            end

        endcase

    end

    always @(\*) begin

        if (state < 4) begin

            case (bcd)

                4'h0:    segout\_1 = 8'hfc;

                4'h1:    segout\_1 = 8'h60;

                4'h2:    segout\_1 = 8'hda;

                4'h3:    segout\_1 = 8'hf2;

                4'h4:    segout\_1 = 8'h66;

                4'h5:    segout\_1 = 8'hb6;

                4'h6:    segout\_1 = 8'hbe;

                4'h7:    segout\_1 = 8'he0;

                4'h8:    segout\_1 = 8'hfe;

                4'h9:    segout\_1 = 8'hf6;

                default: segout\_1 = 8'h00;

            endcase

            segout\_2 = 8'h00;

        end else begin

            case (bcd)

                4'h0:    segout\_2 = 8'hfc;

                4'h1:    segout\_2 = 8'h60;

                4'h2:    segout\_2 = 8'hda;

                4'h3:    segout\_2 = 8'hf2;

                4'h4:    segout\_2 = 8'h66;

                4'h5:    segout\_2 = 8'hb6;

                4'h6:    segout\_2 = 8'hbe;

                4'h7:    segout\_2 = 8'he0;

                4'h8:    segout\_2 = 8'hfe;

                4'h9:    segout\_2 = 8'hf6;

                default: segout\_2 = 8'h00;

            endcase

            segout\_1 = 8'h00;

        end

    end

endmodule

Top顶层模块:

`timescale 1ns / 1ps

module top (

    input        clk,

    input  [3:0] data1,

    input  [3:0] data2,

    output [7:0] an,

    output [7:0] mul,

    output [7:0] result

);

    reg  reset = 0;

*// 时钟信号*

*// reg        clk = 0;*

    wire clk1;

    CLK\_DIV u\_CLK\_DIV (

        .clk0(clk),

        .clk (clk1)

    );

    wire [7:0] z;

*// 乘法器模块*

    MULTU\_4bit u\_MULTU\_4bit (

        .clk  (clk1),

        .reset(reset),

        .a    (data1),

        .b    (data2),

        .z    (z)

    );

*// 将 data1, data2, result 转换为 8421BCD 码*

    wire [ 7:0] data1\_bcd;

    wire [ 7:0] data2\_bcd;

    wire [15:0] z\_bcd;

    Bit4\_to\_8421 u\_Bit4\_to\_8421\_1 (

        .a  (data1),

        .bcd(data1\_bcd)

    );

    Bit4\_to\_8421 u\_Bit4\_to\_8421\_2 (

        .a  (data2),

        .bcd(data2\_bcd)

    );

    Bit8\_to\_8421 u\_Bit8\_to\_8421 (

        .a  (z),

        .bcd(z\_bcd)

    );

*// 数码管显示被乘数、乘数和结果*

    ShowEightSeg7 u\_ShowEightSeg7 (

        .clk     (clk1),

        .seg0    (data1\_bcd[7:4]),

        .seg1    (data1\_bcd[3:0]),

        .seg2    (data2\_bcd[7:4]),

        .seg3    (data2\_bcd[3:0]),

        .seg4    (z\_bcd[15:12]),

        .seg5    (z\_bcd[11:8]),

        .seg6    (z\_bcd[7:4]),

        .seg7    (z\_bcd[3:0]),

        .an      (an),

        .segout\_1(mul),

        .segout\_2(result)

    );

    initial begin

        reset = 1;

    end

endmodule

四、实验结果分析

1.这里应给出相应的实验结果。分析应有条理，要求采用规范的书面语。

2.每个实验都需要做模拟仿真，需要对仿真波形进行简单的文字说明。

3.对下载到开发板上的图片结果做分析说明。

4.原则上要求使用图片与文字结合的形式说明，因为word和PDF文档不支持视频，所以请不要使用视频文件。

5.图片请在垂直方向，不要横向。不要用很大的图片（不要超过1M），请先做裁剪操作。

①激励文件:

4位乘法器:

`timescale 1ns / 1ps

module tb\_MULTU\_4bit;

    parameter XH = 2022217587;//学号

*// Inputs*

    reg        clk = 0;

    reg        reset = 0;

    reg  [3:0] a;

    reg  [3:0] b;

*// Outputs*

    wire [7:0] z;

*// 模拟使用时钟信号*

    always begin

        #(XH / 2) clk <= ~clk;

    end

*// 实例化 MULTU\_4bit 模块*

    MULTU\_4bit u\_MULTU\_4bit (

        .clk  (clk),

        .reset(reset),

        .a    (a),

        .b    (b),

        .z    (z)

    );

*// 初始化*

    initial begin

        reset = 1;

        #XH;

        a <= 4'b0011;

        b <= 4'b0001;

        #XH;

        a <= 4'b0100;

        b <= 4'b0100;

        #XH;

        a <= 4'b0000;

        b <= 4'b0000;

        #XH;

        a <= 4'b0010;

        b <= 4'b1000;

        #XH;

        a <= 4'b1111;

        b <= 4'b0001;

        #XH;

    end

endmodule

16位乘法器:

`timescale 1ns / 1ps

module MULTUtb();

  reg clk;

  reg reset;

  reg [15:0] a;

  reg [15:0] b;

  wire [31:0] z;

*// 实例化被测试的 MULTU 模块*

  MULTU dut (

    .clk(clk),

    .reset(reset),

    .a(a),

    .b(b),

    .z(z)

  );

*// 时钟生成*

  always #5 clk = ~clk;

*// 初始化*

  initial begin

    clk = 0;

    reset = 1;

    #10 reset = 0;  *// 复位信号保持为低电平一段时间后释放*

    #10;

*// 输入值设置*

    a = 16'b0;  *// 设置被乘数*

    b = 16'b0;   *// 设置乘数*

    #10

    a = 16'd7;

    b = 16'd8;

    #10

    a = 16'd9;

    b = 16'd9;

    #10

    a = 16'd3;

    b = 16'd2;

    #10;

*// 输入值设置*

    a = 16'b0;  *// 设置被乘数*

    b = 16'b1111111111111111;   *// 设置乘数*

    #10;

*// 输入值设置*

    a = 16'b1010101010101010;  *// 设置被乘数*

    b = 16'b0;   *// 设置乘数*

    #10;

*// 输入值设置*

    a = 16'b1111111111111111;  *// 设置被乘数*

    b = 16'b1111111111111111;   *// 设置乘数*

    #10;

*// 输入值设置*

    a = 16'b1000000000000000;  *// 设置被乘数*

    b = 16'b1010101010101010;   *// 设置乘数*

    #10;

*// 输入值设置*

    a = 16'b1010101010101010;  *// 设置被乘数*

    b = 16'b1000000000000000;   *// 设置乘数*

    #10;

*// 输入值设置*

    a = 16'b1011110110111101;  *// 设置被乘数*

    b = 16'b1101000011010000;   *// 设置乘数*

    #10;

*// 输入值设置*

    a = 16'b1000111111110001;  *// 设置被乘数*

    b = 16'b1110111001111110;   *// 设置乘数*

    $finish;

  end

endmodule

②约束文件:

*# ============================== 时钟 ============================== #*

set\_property -dict {PACKAGE\_PIN P17 IOSTANDARD LVCMOS33} [get\_ports {clk}]

*# ============================== 拨码开关sw0~sw7 ============================== #*

set\_property -dict {PACKAGE\_PIN P5 IOSTANDARD LVCMOS33} [get\_ports {data1[3]}]

set\_property -dict {PACKAGE\_PIN P4 IOSTANDARD LVCMOS33} [get\_ports {data1[2]}]

set\_property -dict {PACKAGE\_PIN P3 IOSTANDARD LVCMOS33} [get\_ports {data1[1]}]

set\_property -dict {PACKAGE\_PIN P2 IOSTANDARD LVCMOS33} [get\_ports {data1[0]}]

set\_property -dict {PACKAGE\_PIN R2 IOSTANDARD LVCMOS33} [get\_ports {data2[3]}]

set\_property -dict {PACKAGE\_PIN M4 IOSTANDARD LVCMOS33} [get\_ports {data2[2]}]

set\_property -dict {PACKAGE\_PIN N4 IOSTANDARD LVCMOS33} [get\_ports {data2[1]}]

set\_property -dict {PACKAGE\_PIN R1 IOSTANDARD LVCMOS33} [get\_ports {data2[0]}]

*# ============================== 数码管位选信号 ============================== #*

set\_property -dict {PACKAGE\_PIN G2 IOSTANDARD LVCMOS33} [get\_ports {an[0]}]

set\_property -dict {PACKAGE\_PIN C2 IOSTANDARD LVCMOS33} [get\_ports {an[1]}]

set\_property -dict {PACKAGE\_PIN C1 IOSTANDARD LVCMOS33} [get\_ports {an[2]}]

set\_property -dict {PACKAGE\_PIN H1 IOSTANDARD LVCMOS33} [get\_ports {an[3]}]

set\_property -dict {PACKAGE\_PIN G1 IOSTANDARD LVCMOS33} [get\_ports {an[4]}]

set\_property -dict {PACKAGE\_PIN F1 IOSTANDARD LVCMOS33} [get\_ports {an[5]}]

set\_property -dict {PACKAGE\_PIN E1 IOSTANDARD LVCMOS33} [get\_ports {an[6]}]

set\_property -dict {PACKAGE\_PIN G6 IOSTANDARD LVCMOS33} [get\_ports {an[7]}]

*# ============================== 数码管段选信号 ============================== #*

set\_property -dict {PACKAGE\_PIN B4 IOSTANDARD LVCMOS33} [get\_ports {mul[7]}]

set\_property -dict {PACKAGE\_PIN A4 IOSTANDARD LVCMOS33} [get\_ports {mul[6]}]

set\_property -dict {PACKAGE\_PIN A3 IOSTANDARD LVCMOS33} [get\_ports {mul[5]}]

set\_property -dict {PACKAGE\_PIN B1 IOSTANDARD LVCMOS33} [get\_ports {mul[4]}]

set\_property -dict {PACKAGE\_PIN A1 IOSTANDARD LVCMOS33} [get\_ports {mul[3]}]

set\_property -dict {PACKAGE\_PIN B3 IOSTANDARD LVCMOS33} [get\_ports {mul[2]}]

set\_property -dict {PACKAGE\_PIN B2 IOSTANDARD LVCMOS33} [get\_ports {mul[1]}]

set\_property -dict {PACKAGE\_PIN D5 IOSTANDARD LVCMOS33} [get\_ports {mul[0]}]

set\_property -dict {PACKAGE\_PIN D4 IOSTANDARD LVCMOS33} [get\_ports {result[7]}]

set\_property -dict {PACKAGE\_PIN E3 IOSTANDARD LVCMOS33} [get\_ports {result[6]}]

set\_property -dict {PACKAGE\_PIN D3 IOSTANDARD LVCMOS33} [get\_ports {result[5]}]

set\_property -dict {PACKAGE\_PIN F4 IOSTANDARD LVCMOS33} [get\_ports {result[4]}]

set\_property -dict {PACKAGE\_PIN F3 IOSTANDARD LVCMOS33} [get\_ports {result[3]}]

set\_property -dict {PACKAGE\_PIN E2 IOSTANDARD LVCMOS33} [get\_ports {result[2]}]

set\_property -dict {PACKAGE\_PIN D2 IOSTANDARD LVCMOS33} [get\_ports {result[1]}]

set\_property -dict {PACKAGE\_PIN H2 IOSTANDARD LVCMOS33} [get\_ports {result[0]}]

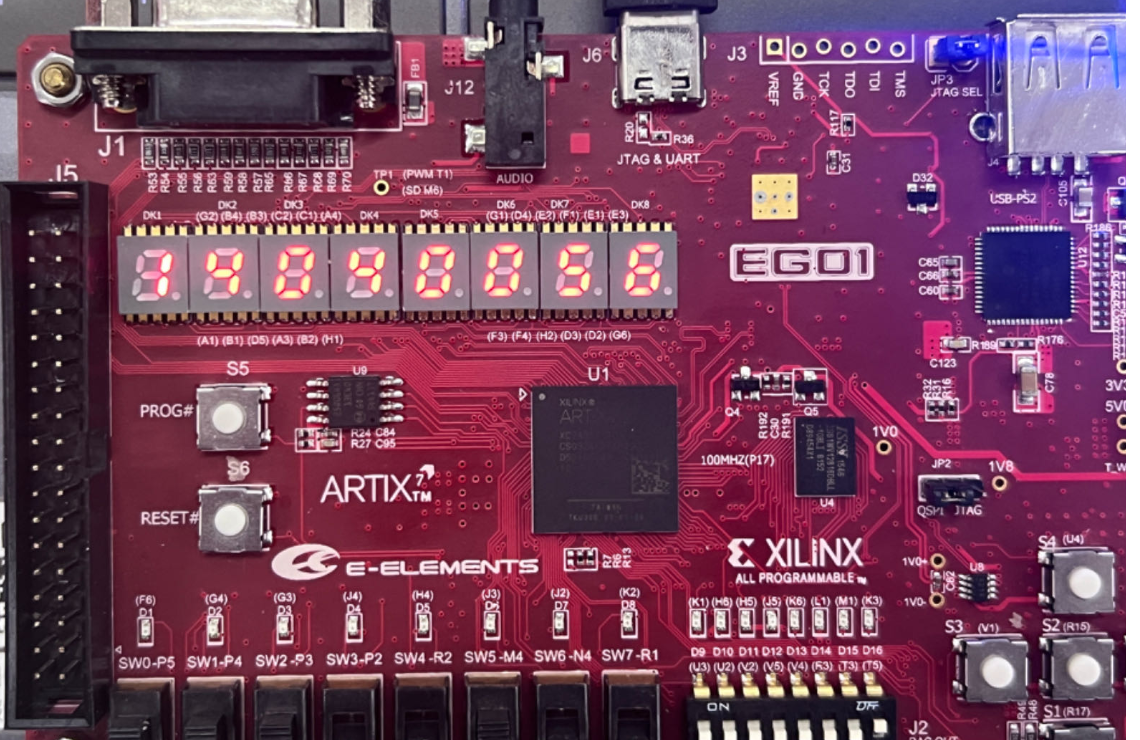
③仿真波形:



④最终结果解释:

在上升沿时候进行计算，在开始时候，rst为1，将数据复位，从第三个上升沿开始计算，计算a\*b=z；0\*0=0；7h\*8h=78h；9h\*9h=51h；3h\*2h=6h；0h\*ffffh=0h；ffffh\*ffffh=fffe0001h；结果均为正确。

⑤开发板运行图片:



五、遇到的问题与实验心得体会

简要地叙述一下实验过程中的感受，以及其他的问题描述和自己的感想。特别是实验中遇到的困难，最后如何解决的。在用verilog代码写程序时遇到语法或其他错误，如何修改解决的。

在计算机组成实验中设计和实现无符号乘法器是一项具有挑战性的任务。在本次实验中，我设计了一个无符号乘法器，用于将两个16位无符号数相乘并得到一个32位无符号数的结果。

在实验过程中，我首先深入理解了无符号乘法的原理和算法。我了解了乘法器的结构和工作原理，并明确了所需的输入和输出信号。在实现的过程中，我参考了指导书中的2个8位无符号数相乘的电路原理图和代码实现，并对其进行了拓展，拓展为2个16位无符号数相乘，因此整个实现过程也相对容易，没有遇到太大的困难。

本次实验中较为复杂的部分是8421码的转换以及数码管的显示，在编写这部分的时候，我通过依次计算每一位实现8421的转换。计算千位时计算输入值a除以1000的结果。由于a是8位的，其值最大为255，因此在实际情况中temp1总是为0。计算百位: 接下来从a中减去temp1 \* 1000（实际上就是a自身，因为temp1为0），然后除以100，得到百位数。之后计算十位: 从a减去千位和百位之和的结果，然后除以10，得到十位数。最后计算个位数，就是从a减去前三个数位计算得到的总和。组合BCD输出 (bcd): 最后，四个临时寄存器的值被组合成一个16位的输出值，格式为{temp1, temp2, temp3, temp4}，每个临时寄存器对应BCD输出的一个十进制位。对于数码管显示的部分，七段显示编码是硬编码在模块内部，对应BCD值从0到9的每个数字的显示方式。例如，4'h0 （BCD的0）对应的编码是 8'hfc，这表示数码管上需要点亮的段来正确显示数字0。

经过本次实验过后，对于乘法器在计算机中的底层实现原理，有了更深的认识。在乘法器的实现中，需要考虑乘法操作的流程、进位的处理和结果的输出。

**实验七 寄存器堆的设计和实现**

一、实验目的及环境

1、深入了解寄存器堆的结构和工作原理。

2、使用Verilog HDL语言来设计和实现寄存器堆结构，进行仿真和下载验证。

3、装有vivado的计算机 1台

4、EGO1开发板 1块

二、实验目标及任务

1、设计由16个4位寄存器构成的寄存器堆，该寄存器堆是双端口输出。进行功能仿真

2、设计由16个4位寄存器构成的寄存器堆，引脚绑定后进行下载，在EGO1开发板上进行数据验证。

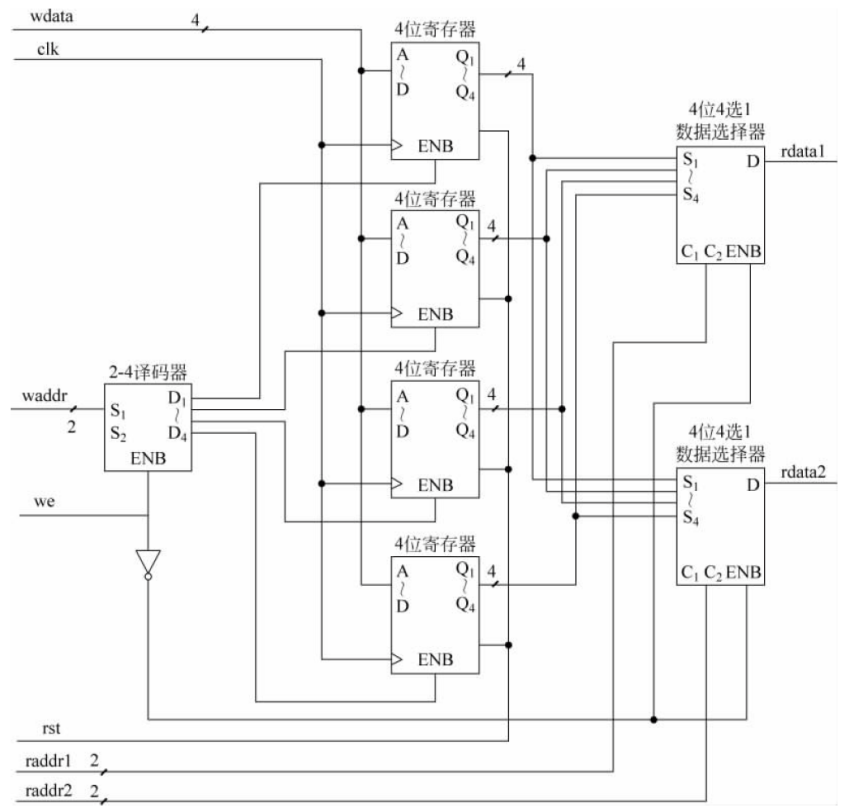
三、实验过程及记录



本节重点介绍实验的具体过程，包括：代码设计层次结构图及说明、源代码（包括注释）、PC机上进行的关键步骤截图及说明、调试过程等，这部分的内容应当与实际操作过程类似即可（简单明了）。

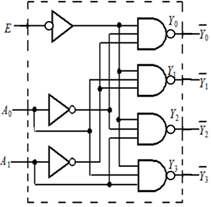
①电路原理图:

寄存器堆:



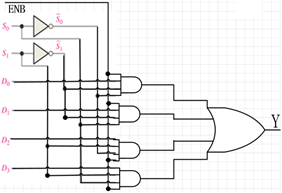
上图是4个4位寄存器构成的寄存器堆的原理图，相似的，我们可以根据这张图得到16个4位寄存器的寄存器堆：添加更多的寄存器至16个，将2-4译码器改为4-16译码器，将4位4选1数据选择器改为4位的16选1数据选择器。接下来的目标为：实现4-16译码器、16选1数据选择器和4位寄存器。

4-16译码器：



我们先观察2-4译码器的原理图，然后将其拓展为一个4-16译码器的原理图，并按照如上原理图来将其实现。

16选1数据选择器:



同样的，先观察4选1数据选择器的原理图，然后将其拓展为一个16选1数据选择器，并按照上述原理图将其中的逻辑门例化。

寄存器：在寄存器中，我们需要在寄存器中先声明一个reg\_data的寄存器类型变量，用于存储该寄存器中的值，然后响应下降沿，根据rst位，执行对应的操作即可。

最后是顶层文件的设计，根据寄存器堆的原理图，确定输入输出信号，然后声明一些中间的信号线，然后根据原理图，将元件一一例化，并将数据线或信号线相连。

四、实验结果分析

1.这里应给出相应的实验结果。分析应有条理，要求采用规范的书面语。

2.每个实验都需要做模拟仿真，需要对仿真波形进行简单的文字说明。

3.对下载到开发板上的图片结果做分析说明。

4.原则上要求使用图片与文字结合的形式说明，因为word和PDF文档不支持视频，所以请不要使用视频文件。

5.图片请在垂直方向，不要横向。不要用很大的图片（不要超过1M），请先做裁剪操作。

五、遇到的问题与实验心得体会

简要地叙述一下实验过程中的感受，以及其他的问题描述和自己的感想。特别是实验中遇到的困难，最后如何解决的。在用verilog代码写程序时遇到语法或其他错误，如何修改解决的。

1、……

**实验八 XXXXXX**

一、实验目的及环境

1.……

二、实验目标及任务

1.

……

2.

……

三、实验过程及记录



本节重点介绍实验的具体过程，包括：代码设计层次结构图及说明、源代码（包括注释）、PC机上进行的关键步骤截图及说明、调试过程等，这部分的内容应当与实际操作过程类似即可（简单明了）。

1、……

四、实验结果分析

1.这里应给出相应的实验结果。分析应有条理，要求采用规范的书面语。

2.每个实验都需要做模拟仿真，需要对仿真波形进行简单的文字说明。

3.对下载到开发板上的图片结果做分析说明。

4.原则上要求使用图片与文字结合的形式说明，因为word和PDF文档不支持视频，所以请不要使用视频文件。

5.图片请在垂直方向，不要横向。不要用很大的图片（不要超过1M），请先做裁剪操作。

五、遇到的问题与实验心得体会

简要地叙述一下实验过程中的感受，以及其他的问题描述和自己的感想。特别是实验中遇到的困难，最后如何解决的。在用verilog代码写程序时遇到语法或其他错误，如何修改解决的。

1、……

|  |
| --- |
| 一、原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  特此声明！  **作者签字: 嵌入签名图片** |
| 二、对课程实验的学术评语（教师填写） |
|  |
| 三、对课程实验的评分（教师填写） |
| |  |  |  |  | | --- | --- | --- | --- | | 评分项目  （分值） | 报告撰写  （50分） | 课设验收  （50分） | 最终评定  （100分） | | 得分 |  |  |  | |
| **指导教师签字:** |

物联网1班和2班提交实验5-8实验报告的链接：

<https://send2me.cn/IuId3Ca1/QGSLhXdskkWmnQ>



计算机1班、2班和3班提交实验5-8实验报告的链接：

https://send2me.cn/zpy-PLTV/QqCUMzwR-3\_EHg



计算机4班和5班提交实验5-8实验报告的链接：

https://send2me.cn/onBsL6Gc/QsWLLqvdK\_3LqQ

