电子科技大学

计算机专业类课程

实验报告

课程名称：计算机组成原理综合实验

学生姓名：郭志猛

学　　号：2017080201005

指导教师：陈虹

日　　期：2019 年　6 月 29 日

目录

电 子 科 技 大 学

实 验 报 告 一

一 实验室名称 A2-411

二 实验项目：实验一：基本器件的设计与实现

三 实验环境：

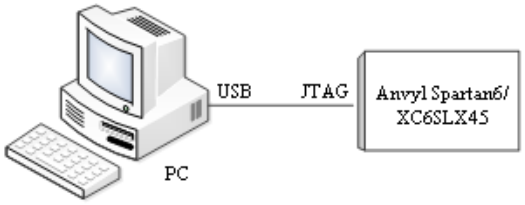
硬件环境：PC 计算机和 FPGA 开发板

软件环境：操作系统：Windows 10

开发平台：Xilinx ISE Design Suite 14.7 集成开发系统

开发板：Spartan6-XC6SLX45

下载软件：Adept



四 实验任务

1. 掌握用 Verlog 设计硬件电路的基本方法；
2. 开发板的基本使用；
3. 基本器件的设计：

   32 位 2 选 1 多路选择器；

   5 位 2 选 1 多路选择器；

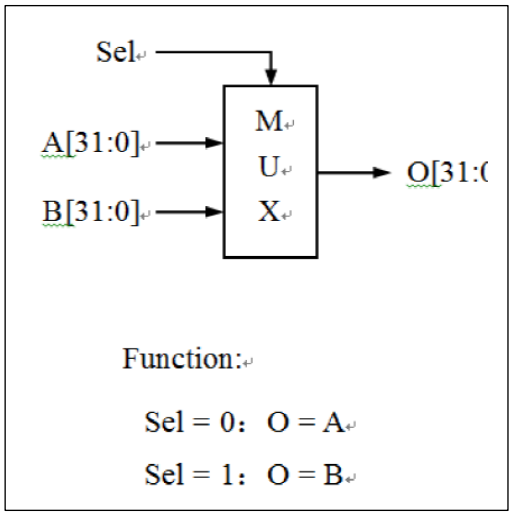
   32 位寄存器堆；

   扩展器与 ALU 的设计。

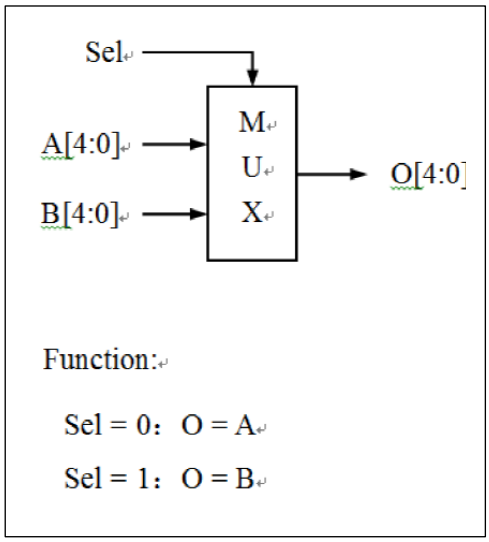
五 实验原理

1. 32 位 2 选 1 多路选择器

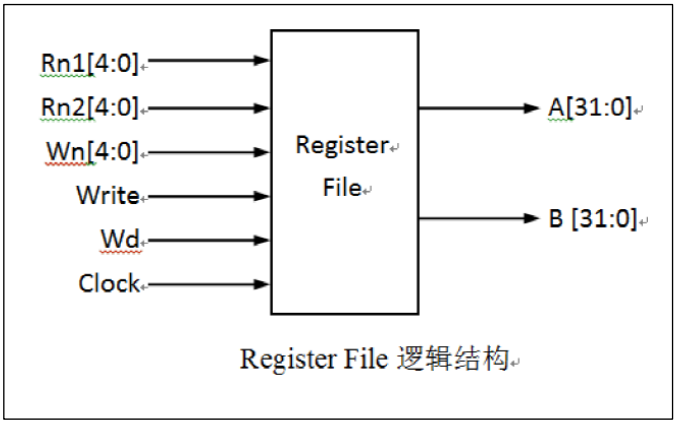
电路设计如下：



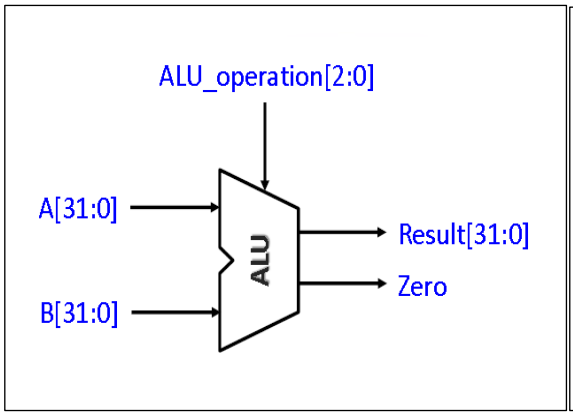
2. 5 位 2 选 1 多路选择器



3. 32 位寄存器堆



4. ALU 的设计



ALU_Operation	功能描述
000	Add（加）
100	Sub（减）
001	And（与）
101	Or（或）
010	Xor（异或）
110	设置高位

六 实验步骤

1. 新建文件：

设置设备和文件路径及编程语言

2. 新建模块：

添加 Verilog Module：

3. 编写程序：

根据上图控制电路功能编写 Verilog 程序：

（5 位 2 选 1 多路选择器）

```
21  module MUX5_2_1(
22      input [4:0] A,
23      input [4:0] B,
24      input       Sel,
25      output[4:0] O
26   );
27
28      assign O = Sel?B:A;
29  endmodule
30
```

（32 位 2 选 1 多路选择器）

```
21  module MUX32_2_1(
22      input [31:0] A,
23      input [31:0] B,
24      input        Sel,
25      output[31:0] O
26     );
27
28        assign O = Sel? B:A;
29  endmodule
30
```

（32 位寄存器堆）

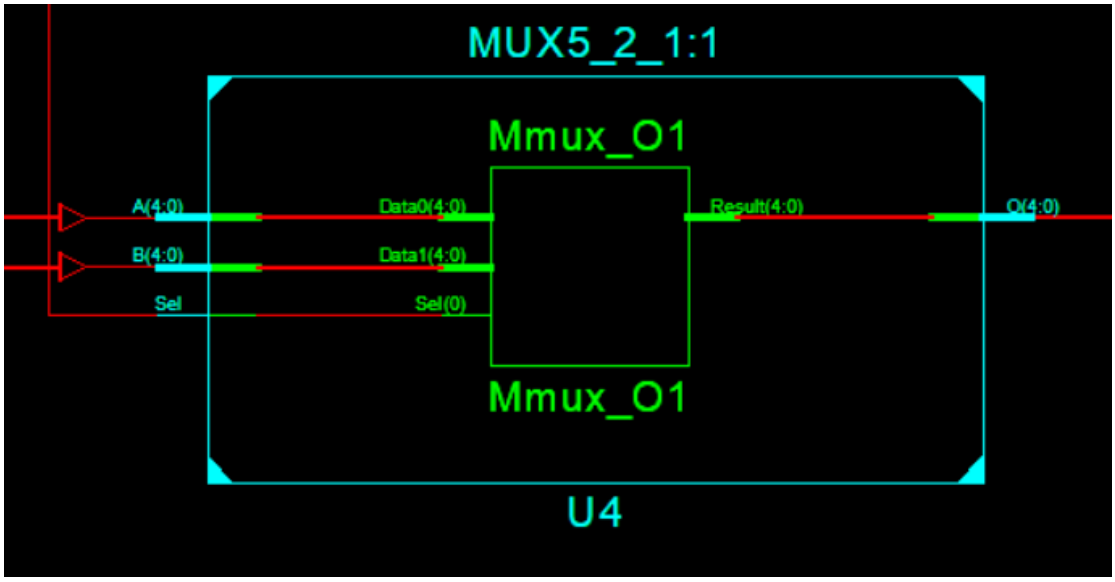
```
module regfile(
         input  [4:0]  Rn1, Rn2, Wn,
         input        Write,
         input  [31:0] Wd,
         output [31:0] A,B,
         input         Clock
      );

      reg [31:0] Register[0:31];

      //read data
      assign A = (Rn1 == 0)? 0:Register[Rn1];
      assign B = (Rn2 == 0)? 0:Register[Rn2];

      //write data
      always @ ( posedge Clock) begin
         if ((Write) && (Wn != 0)) Register[Wn] <= Wd;
      end

endmodule
```

（ALU）

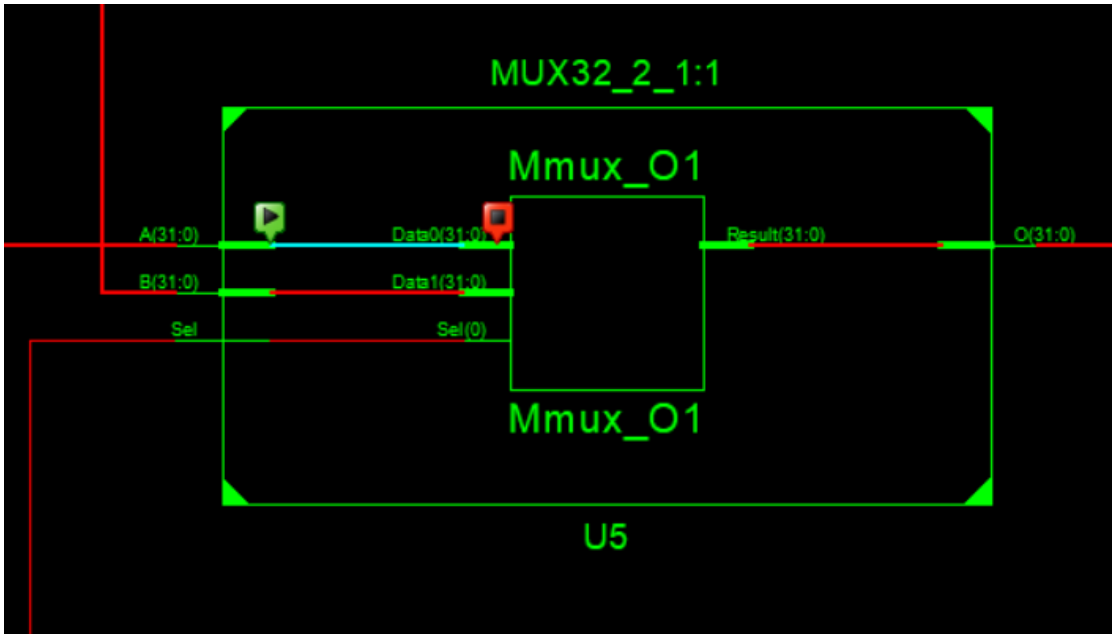
```
module ALU(
    input [31:0] A,B,
    input [ 2:0] ALU_operation,
    output[31:0] Result,
    output       Zero
    );

    assign Result = (ALU_operation == 3'b000) ? A + B :
                    (ALU_operation == 3'b100) ? A - B :
                    (ALU_operation == 3'b001) ? A & B :
                    (ALU_operation == 3'b101) ? A | B :
                    (ALU_operation == 3'b010) ? A ^ B :
                    (ALU_operation == 3'b110) ? {B[15:0],16'h0} :
                    32'hxxxxxxxx;

    assign Zero = ~|Result;


endmodule

```

4. 综合模块：

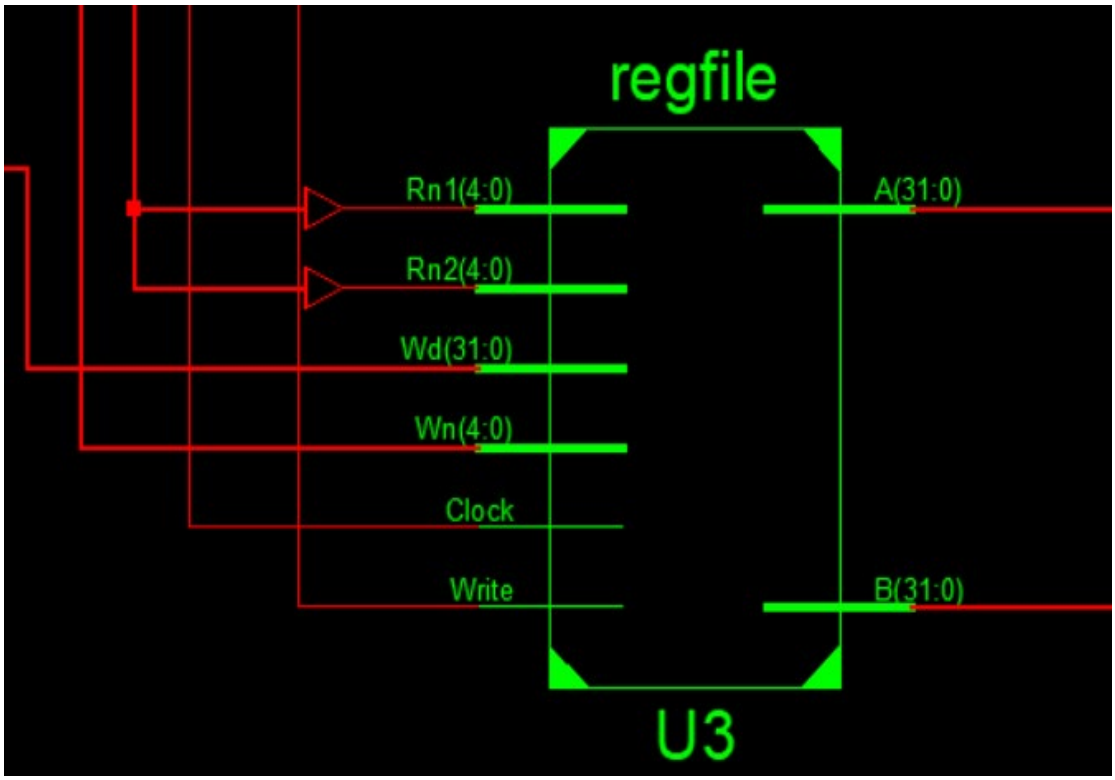
（5 位 2 选 1 多路选择器）



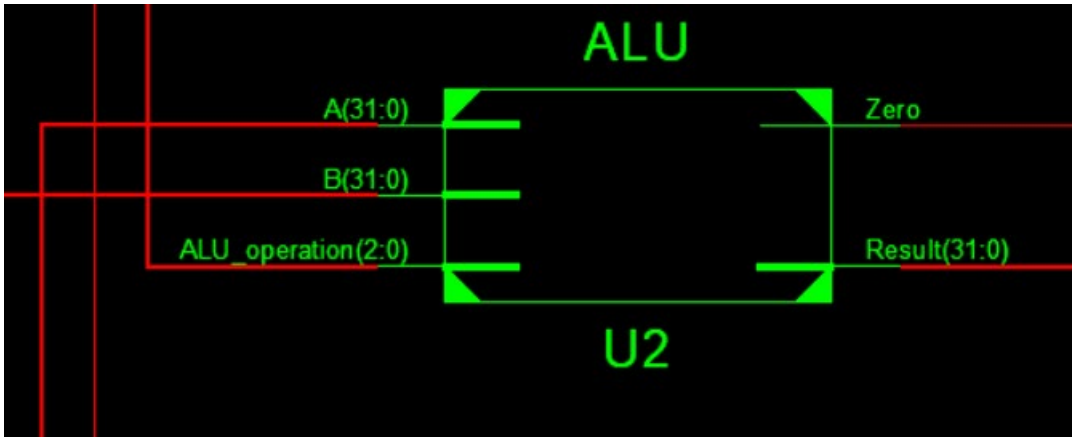
（32 位 2 选 1 多路选择器）



(32 位寄存器堆)



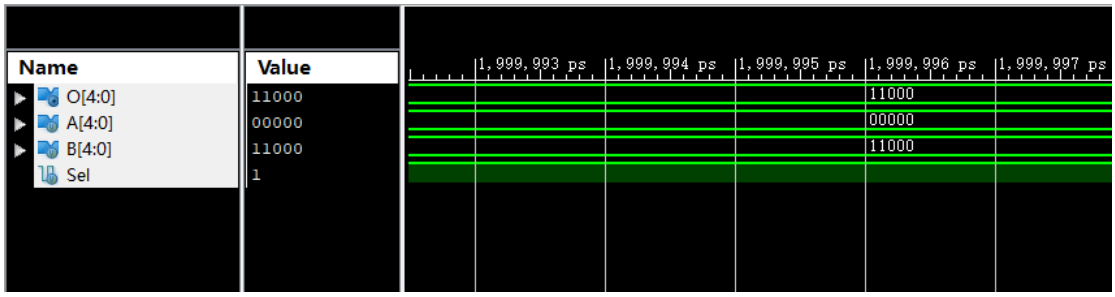
(ALU)



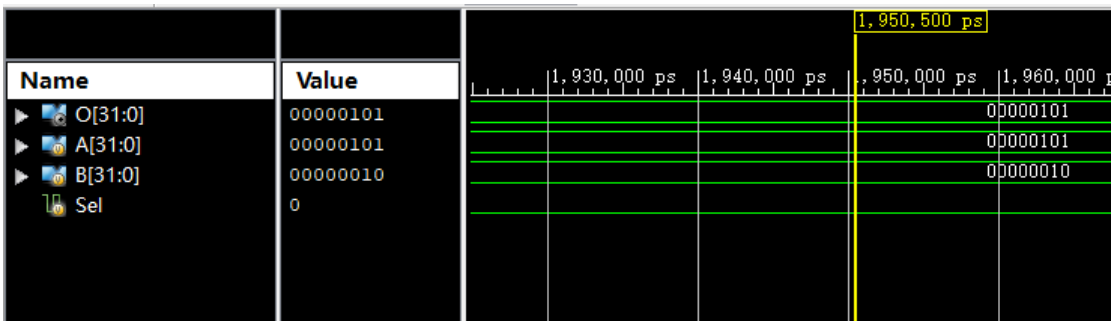
七 实验结果

实验结果如下：

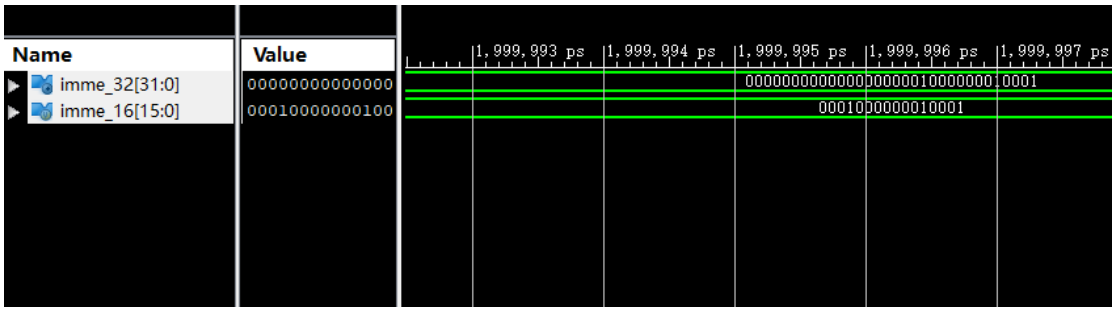
（5 位 2 选 1 多路选择器）



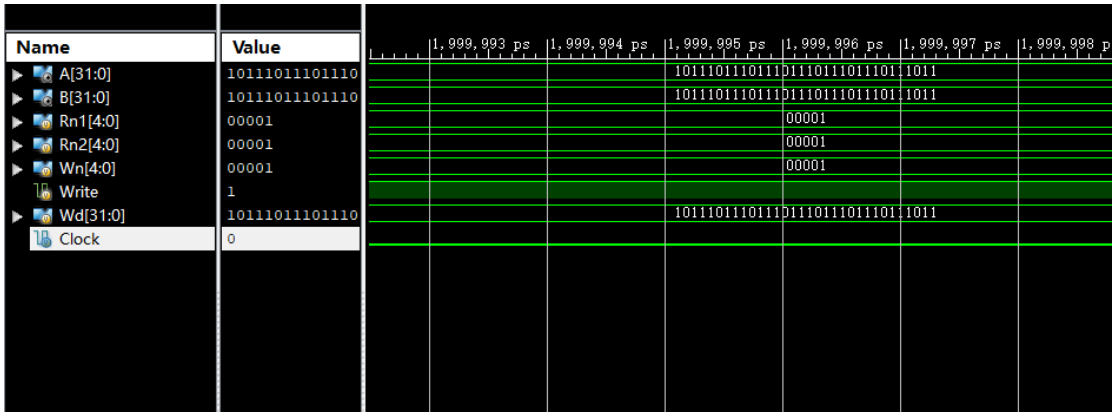
（32 位 2 选 1 多路选择器）


Name
Value
O[31:0]
A[31:0]
B[31:0]
Sel
00000101
00000101
00000010
0
1,950,500 ps
1,930,000 ps
1,940,000 ps
1,950,000 ps
1,960,000 p
00000101
00000101
00000010

（32 位寄存器堆）


Name
Value
imme_32[31:0]
imme_16[15:0]
00000000000000
00010000000100
1,999,993 ps
1,999,994 ps
1,999,995 ps
1,999,996 ps
1,999,997 ps
00000000000000000010000000010001
00010000000010001

（ALU）


Name
Value
A[31:0]
B[31:0]
Rn1[4:0]
Rn2[4:0]
Wn[4:0]
Write
Wd[31:0]
Clock
10111011101110
10111011101110
00001
00001
00001
1
10111011101110
0
1,999,993 ps
1,999,994 ps
1,999,995 ps
1,999,996 ps
1,999,997 ps
1,999,998 p
10111011101110111011101110111011
10111011101110111011101110111011
00001
00001
00001
10111011101110111011101110111011

电 子 科 技 大 学

实 验 报 告 二

一 实验室名称 A2-411

二 实验项目：实验二：控制器与取指电路的设计与实现

三 实验环境：

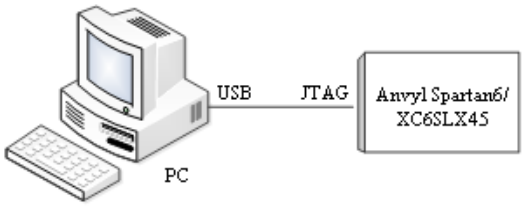
硬件环境：PC 计算机和 FPGA 开发板

软件环境：操作系统：Windows 10

开发平台：Xilinx ISE Design Suite 14.7 集成开发系统

开发板：Spartan6-XC6SLX45

下载软件：Adept



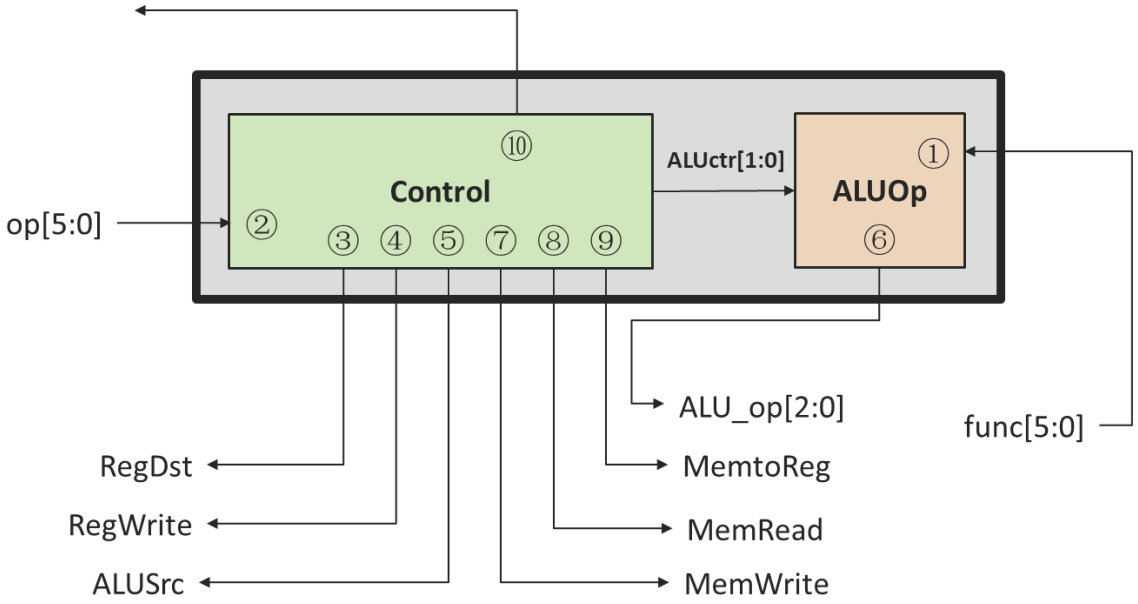
四 实验任务

1. 设计并实现控制器
2. 设计并实现取指电路

五 实验原理

1. 控制器的设计

控制器结构如下图所示：

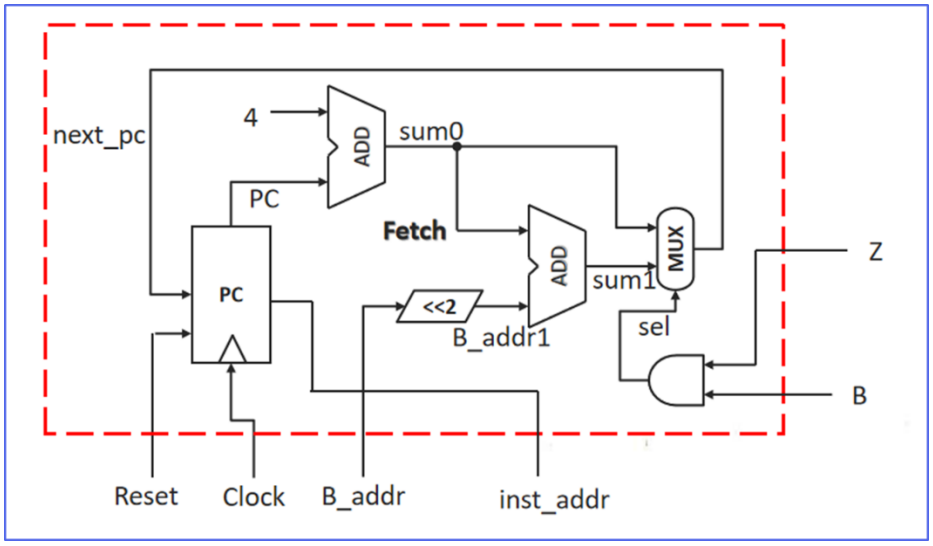


输入指令与输出控制信号关系如下：

Input		Ouppput							
指令	op[5:0]	RegDst	RegWrite	ALUSrc	MemWrite	MemRead	MemtoReg	Branch	ALUctr[1:0]
RT	000000	1	1	0	0	0	0	0	10
lw	100011	0	1	1	0	1	1	0	00
sw	101011	x	0	1	1	0	x	0	00
beq	000100	x	0	0	0	0	x	1	01
lui	001111	0	1	1	0	0	0	0	11

2. 取指电路的设计

取指电路如下图所示：



六 实验步骤

1.新建文件：

设置设备和文件路径及编程语言

2.新建模块：

添加 Verilog Module

3.编写程序：

根据上图控制电路功能编写 Verilog 程序

（控制器：Control+ALUop）

```
21  module Control(
22      input [5:0] op,
23      output RegDst,RegWrite,ALUSrc,
24      output MemWrite,MemRead,MemtoReg,
25      output Branch,
26      output [1:0] ALUctr
27      );
28
29      wire i_Rt = ~|op;
30      wire i_lw = op[5]&~op[3];
31      wire i_sw = op[5]&op[3];
32      wire i_beq = ~op[3]&op[2];
33      wire i_lui = op[3]&op[2];
34
35      assign RegDst = i_Rt;
36      assign RegWrite = i_Rt | i_lw | i_l
37      assign ALUSrc = i_lw|i_sw|i_lui;
38      assign MemWrite = i_sw;
39      assign MemRead = i_lw;
40      assign MemtoReg = i_lw;
41      assign Branch = i_beq;
42
43      assign ALUctr[1] = i_Rt|i_lui;
44      assign ALUctr[0] = i_beq|i_lui;
45
46
47  endmodule
```

```
22  module ALUop(
23     input [5:0] func,
24     input [1:0] ALUctr,
25     output [2:0] ALU_op
26     );
27
28     wire i_Rt = ALUctr[1] & ~ALUctr[0];
29
30     assign ALU_op[2] = (i_Rt & ((~func[2] & func[1])|(func[2]&func[0])))|ALUctr[0];
31     assign ALU_op[1] = (i_Rt&func[1]&func[2])|(ALUctr[0]&ALUctr[1]);
32     assign ALU_op[0] = (i_Rt&func[2]&~func[1]);
33
34  endmodule
```

（取指电路：总+加法器+左移两位+32 位多路选择器）

```
21  module Fetch(
22     input B,Z,Reset,Clock,
23     input [31:0] B_addr,
24     output [31:0] addr
25     );
26
27     reg [31:0] PC;
28     wire [31:0] sum0,B_addr1,sum1,next_pc;
29     wire sel = Z&B;
30
31     Left_2_Shifter U0(B_addr,B_addr1);
32
33     ADD32 U1(PC,4,sum0);
34     ADD32 U2(sum0,B_addr1,sum1);
35     MUX32_2_1 M1(sum0,sum1,sel,next_pc);
36
37     assign addr = PC;
38
39     always @(posedge Clock or negedge Reset) begin
40     if(Reset == 0)
41        PC = 0;
42     else
43        PC = next_pc;
44     end
45
46
47
48  endmodule
```

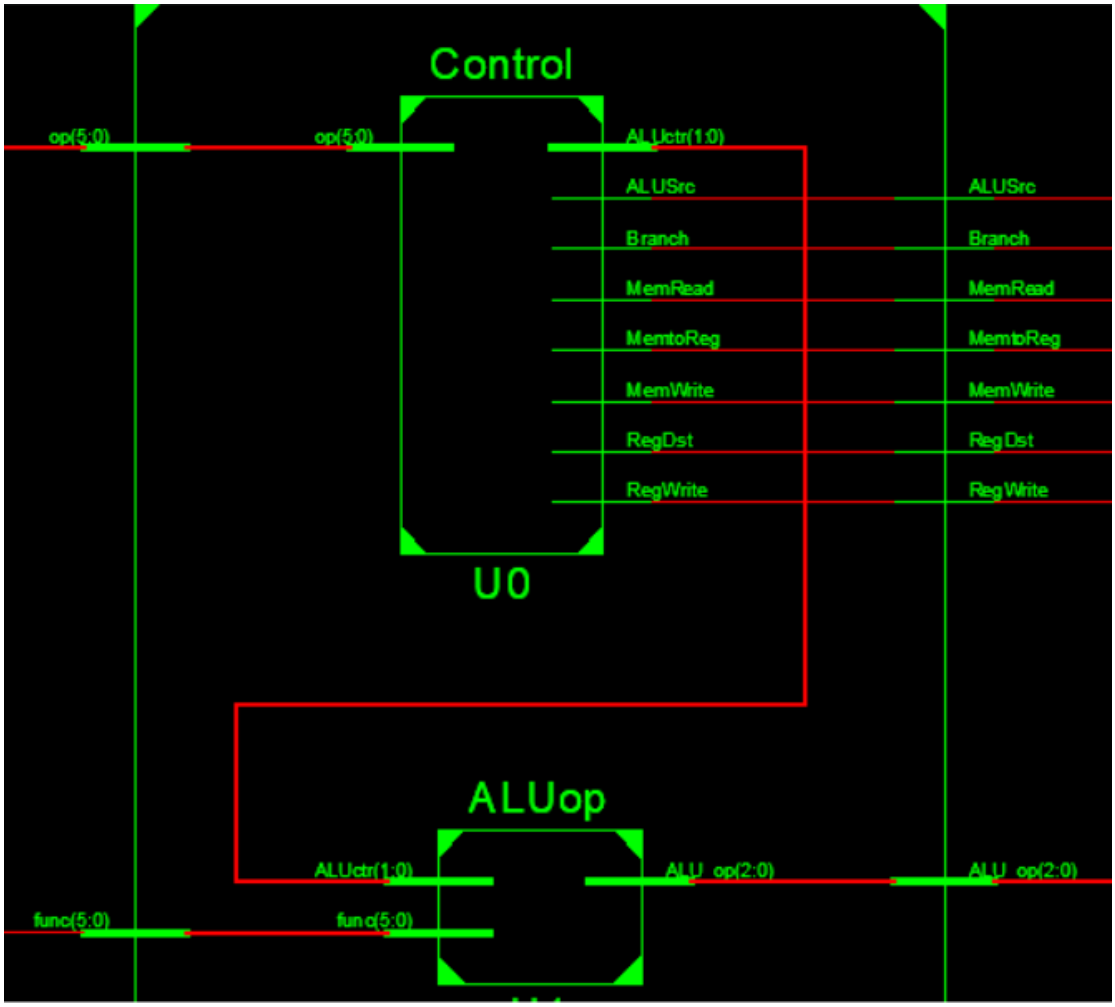
```
21  module ADD32(A,B,C);
22      input [31:0] A,B;
23      output [31:0] C;
24
25      assign C = A+B;
26
27
28  endmodule
```

```
21  module Left_2_Shifter(B_addr, B_addr1);
22     input     [31:0] B_addr;
23     output    [31:0] B_addr1;
24
25   assign  B_addr1[31:0] = { B_addr[29:0], 2'b00 };
26
27  endmodule
```

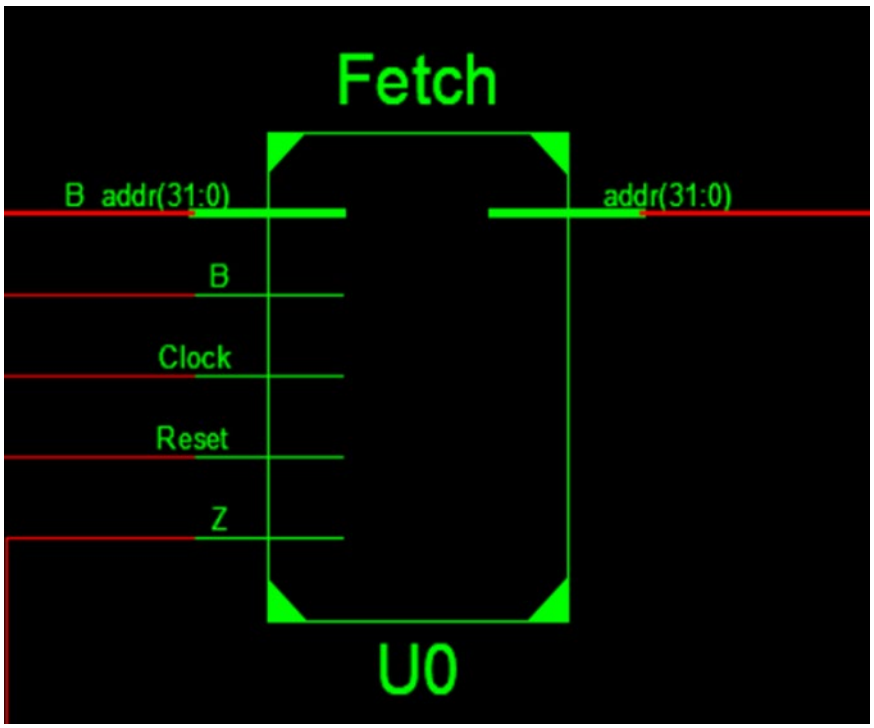
```
21  module MUX32_2_1(
22       input [31:0] A,
23       input [31:0] B,
24       input        Sel,
25       output[31:0] O
26      );
27
28         assign O = Sel? B:A;
29  endmodule
```

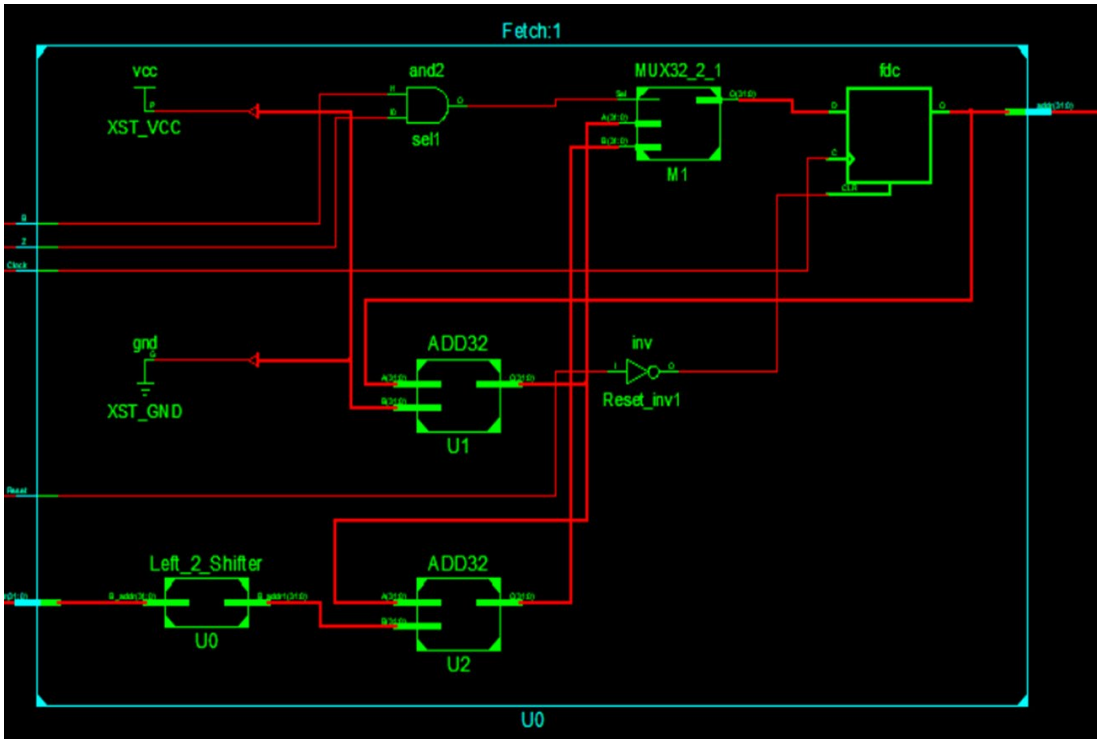
4. 综合模块：

（控制器）


Control
op(5:0)
op(5:0)
ALUctr(1:0)
ALUSrc
Branch
MemRead
MemtoReg
MemWrite
RegDst
RegWrite
ALUSrc
Branch
MemRead
MemtoReg
MemWrite
RegDst
RegWrite
U0
ALUop
ALUctr(1:0)
ALU_op(2:0)
ALU_op(2:0)
func(5:0)
func(5:0)

（取指电路）

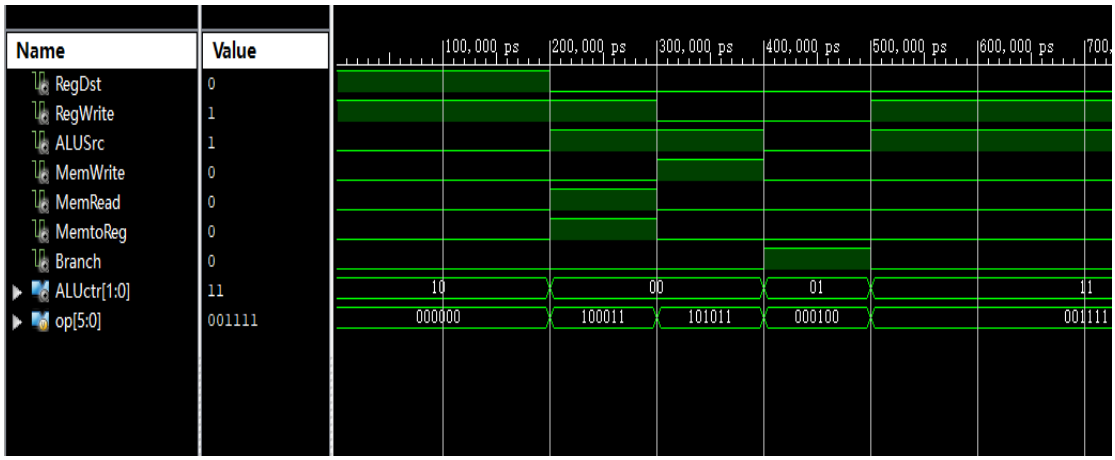

Fetch
B_addr(31:0)
addr(31:0)
B
Clock
Reset
Z
U0



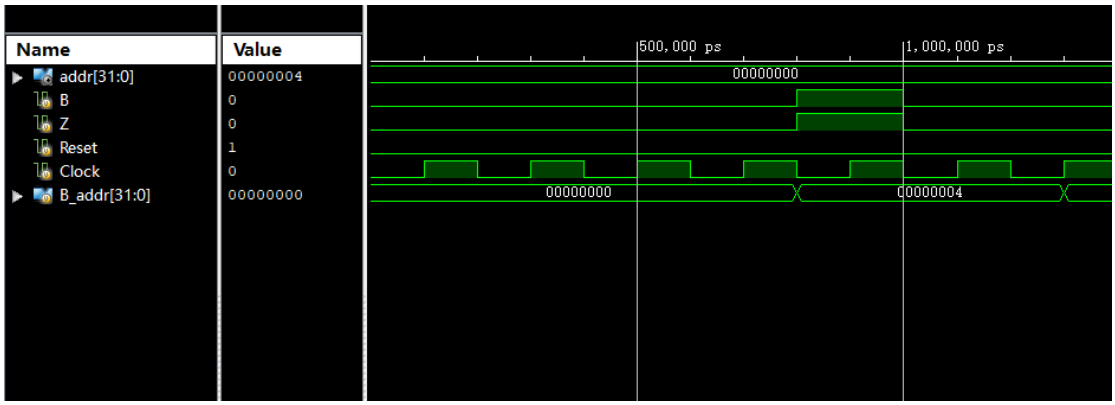
七 实验结果

实验结果如下：

（控制器）



（取指电路）


Name
Value
addr[31:0]
00000004
B
0
Z
0
Reset
1
Clock
0
B_addr[31:0]
00000000
500,000 ps
1,000,000 ps
00000000
00000000
00000004

电 子 科 技 大 学

实 验 报 告 三

一 实验室名称 A2-411

二 实验项目：实验三：单周期 CPU 的设计与实现

三 实验环境：

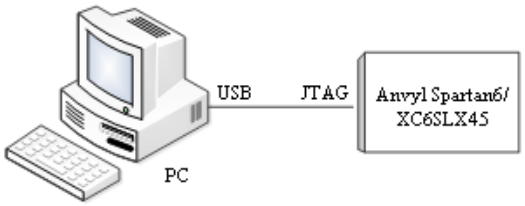
硬件环境：PC 计算机和 FPGA 开发板

软件环境：操作系统：Windows 10

开发平台：Xilinx ISE Design Suite 14.7 集成开发系统

开发板：Spartan6-XC6SLX45

下载软件：Adept



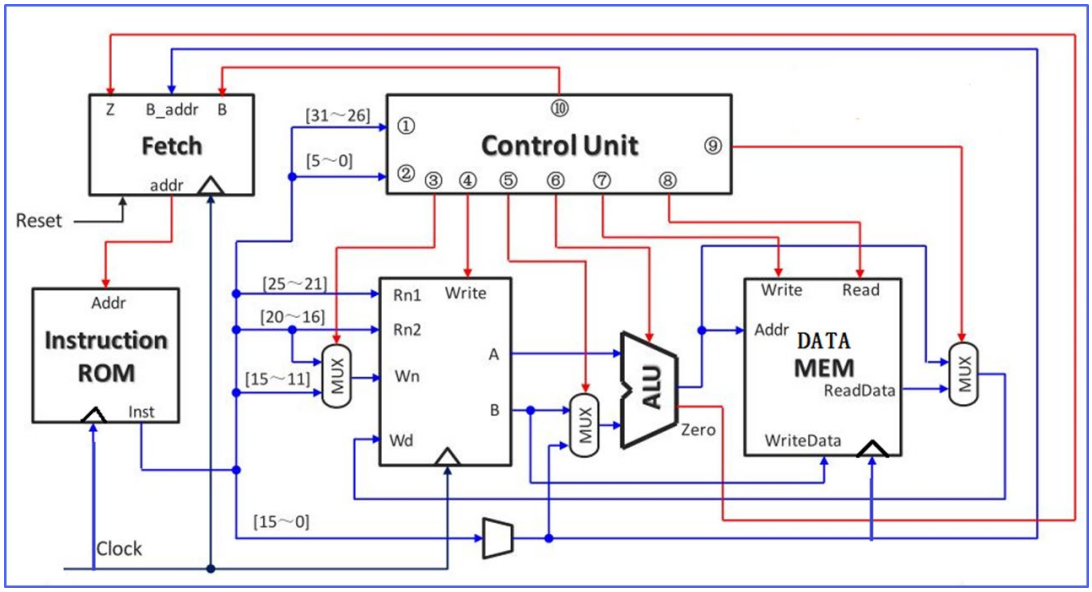
四 实验任务

1. 设计并实现 Data_Flow
2. 设计并实现 Instruction_ROM
3. 设计并实现 Data_MEM

五 实验原理

1. Data_Flow 的设计

三个部分的结构如下图所示：



Data_Flow 的输入输出信号如下：

输入信号：

1. Reset—复位信号
2. Clock—时钟信号
3. Inst—从指令存储器读出的指令码（32位）
4. Data— Data Mem的输出作为最后一个MUX的输入数据（32位）

输出信号：

1. MemWrite—数据存储器DATA MEM的写信号
2. MemRead—数据存储器DATA MEM的读信号
3. Result—运算器ALU的输出结果（32位）
4. B_data—寄存器堆B口输出作为数据存储器DATA MEM的写入数据（32位）
5. NextPC—取指电路形成的下条指令的地址（32位）

2. Instruction_ROM 的设计

Instruction_ROM 输入输出信号如下：

输入信号：Addr —地址（32位）

输出信号：Inst — 当前指令（32位）

MIPS32 常见指令编码表如下：

指令	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]	功能
add	000000	rs	rt	rd	00000	100000	寄存器加
sub	000000	rs	rt	rd	00000	100010	寄存器减
and	000000	rs	rt	rd	00000	100100	寄存器与
or	000000	rs	rt	rd	00000	100101	寄存器或
xor	000000	rs	rt	rd	00000	100110	寄存器异或

指令	[31:26]	[25:21]	[20:16]	[15:0]	功能
lw	100011	rs	rt	immediate	取字数据
sw	101011	rs	rt	immediate	存字数据
beq	000100	rs	rt	immediate	相等转移
lui	001111	00000	rt	immediate	设置高位

需要完成的指令如下：

```
Add $a1, $0, $0
Lw   $s1, 0($a1)
Lw   $s2, 4($a1)
Add $s1, $s1, $s2
Lw   $s2, 8($a1)
Beq $s1, $s2, 1
Sw   $0, 12($a1)
 Sw   $s1, 12($a1)
```

3.Data_MEM 的设计

Data_MEM 的输入输出信号如下：

输入信号：Addr —地址（32位）

Read,Write — 读、写控制信号（1位）

DataIn — 要写入地址指示单元的数据（32位）

Clock — 时钟

输出信号：DataOut— 地址指示单元的输出数据（32位）

六 实验步骤

1.新建文件：

设置设备和文件路径及编程语言

2.新建模块：

添加 Verilog Module

3.编写程序：

根据上图控制电路功能编写 Verilog 程序

（Data_Flow）

```
21  module data_flow(
22      input reset,
23      input clk,
24      input [31:0] inst,
25      input [31:0] data,
26      output memwrite,
27      output memread,
28      output [31:0] result,
29      output [31:0] B_data,
30      output [31:0] nextpc
31      );
32
33      wire [31:0] B_addr;
34      wire Z,B;
35      wire regdst;
36      wire regwrite;
37      wire ALUsrc;
38      wire memtoreg;
39      wire [2:0] ALU_op;
40      wire [31:0] ALU_A,ALU_B;
41      wire [4:0] wn;
42      wire [31:0] wd;
```

```
45    Fetch U0(B,Z,reset,clk,B_addr,nextpc);
46    Control_Unit U1(inst[31:26],inst[5:0],regdst,regwrite,ALUsrc,memwrite,memread,memtoreg,B,ALU_op);
47    ALU U2(ALU_A,ALU_B,ALU_op,result,Z);
48    regfile U3(inst[25:21],inst[20:16],wn,regwrite,wd,ALU_A,B_data,clk);
49    MUX5_2_1 U4(inst[20:16],inst[15:11],regdst,wn);
50    MUX32_2_1 U5(B_data,B_addr,ALUsrc,ALU_B);
51    extender U6(inst[15:0],B_addr);
52    MUX32_2_1 U7(nextpc,data,memtoreg,wd);
53
54  endmodule
```

（Instruction MEM）

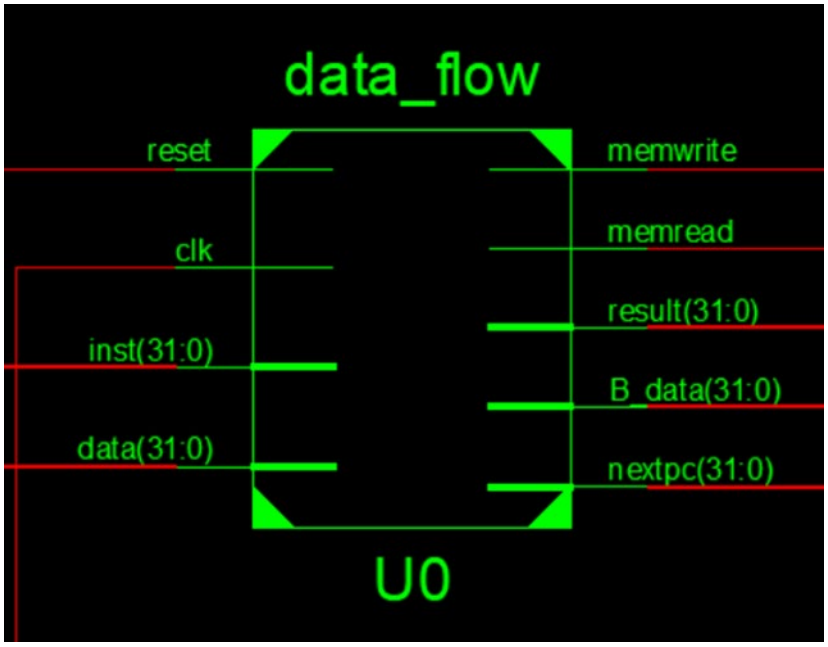
```
21  module inst_mem(
22      input  [31:0] addr,
23      output [31:0] inst
24  );
25      wire [31:0] ram [0:31];
26
27     assign ram[0] = 32'h00002820;
28     assign ram[1] = 32'h8cb10000;
29     assign ram[2] = 32'h8ca10004;
30     assign ram[3] = 32'h00220820;
31     assign ram[4] = 32'h8ca10008;
32     assign ram[5] = 32'h8ca10008;
33     assign ram[6] = 32'haca0000c;
34     assign ram[7] = 32'haca1000c;
35     assign inst = ram[addr[6:2]];
36
37
38  endmodule
```

（Data_MEM）

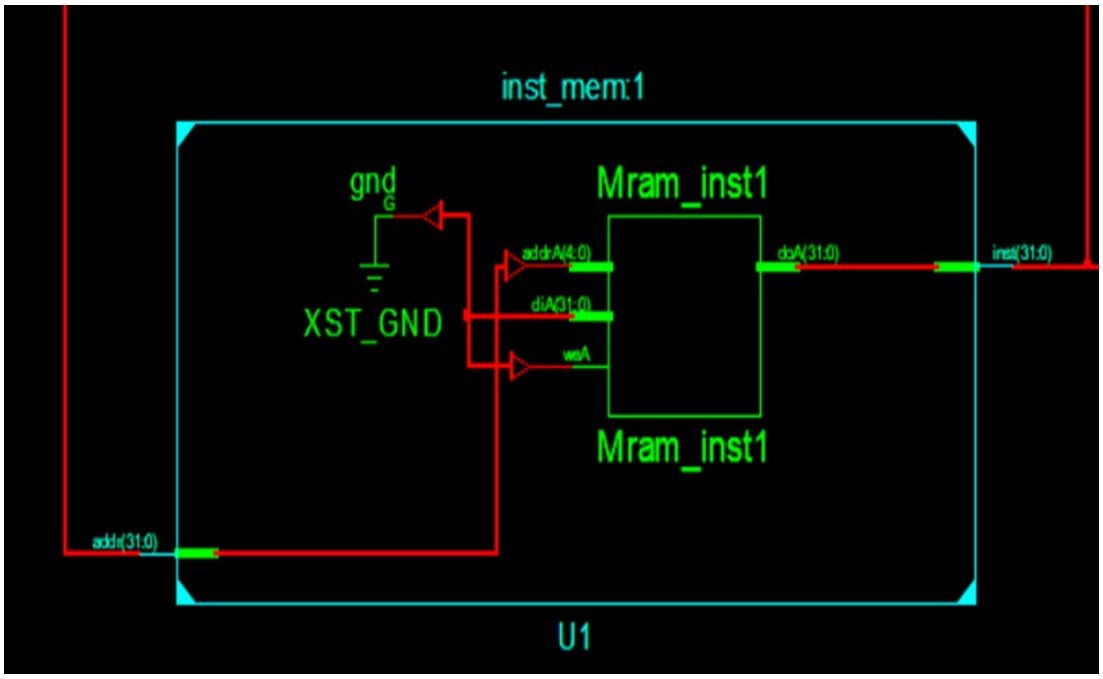
```
21  module Data_Mem( Addr,Read,Write,DataIn,Clock,DataOut);
22     input [31:0] Addr;
23     input Read,Write;
24     input [31:0] DataIn;
25     input Clock;
26     output [31:0] DataOut;
27
28     reg [31:0] ram[0:31];
29     assign DataOut = Read?ram[Addr[6:2]]:32'hxxxxxxxx;
30
31     always @ (posedge Clock) begin
32        ram[Addr[6:2]] = Write?DataIn:32'hxxxxxxxx;
33     end
34     integer i;
35
36     initial begin
37        for(i=0;i<32;i=i+1)
38        ram[i] = i;
39     end
40
41
42  endmodule
```

4.综合模块：

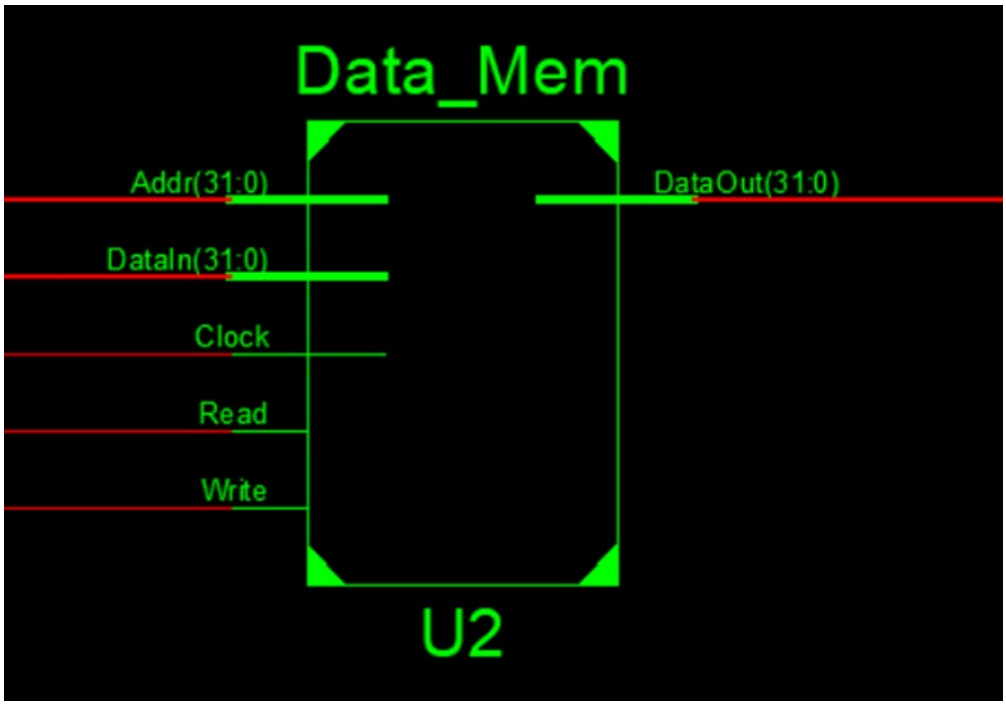
（Data_Flow）



（Instruction_MEM）



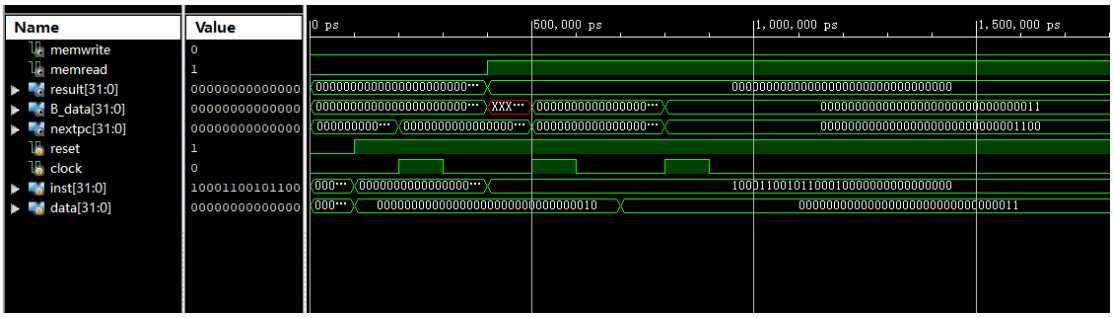
（Data_MEM）



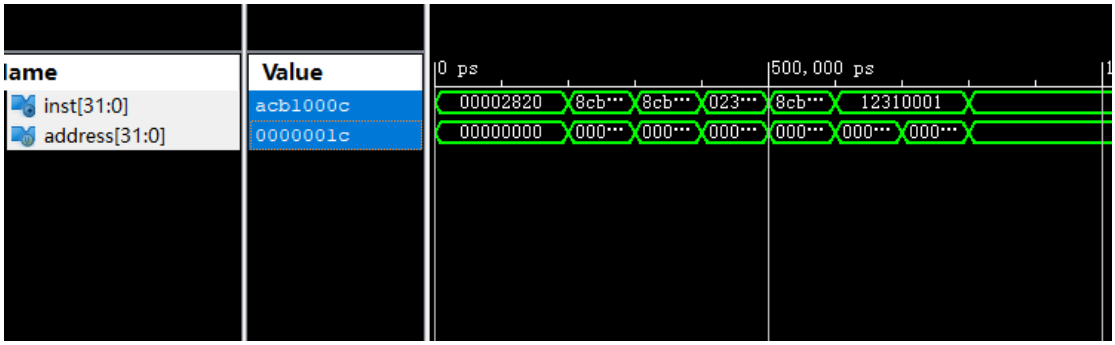
七 实验结果

实验结果如下：

（Data_Flow）



（Instruction_MEM）



电 子 科 技 大 学

实 验 报 告 四

一 实验室名称 A2-411

二 实验项目：实验四：单周期计算机的设计与实现

三 实验环境：

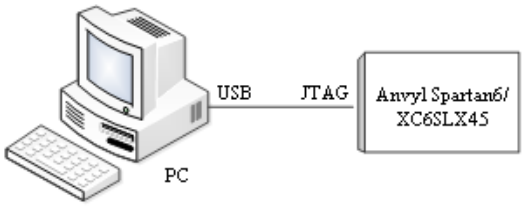
硬件环境：PC 计算机和 FPGA 开发板

软件环境：操作系统：Windows 10

开发平台：Xilinx ISE Design Suite 14.7 集成开发系统

开发板：Spartan6-XC6SLX45

下载软件：Adept



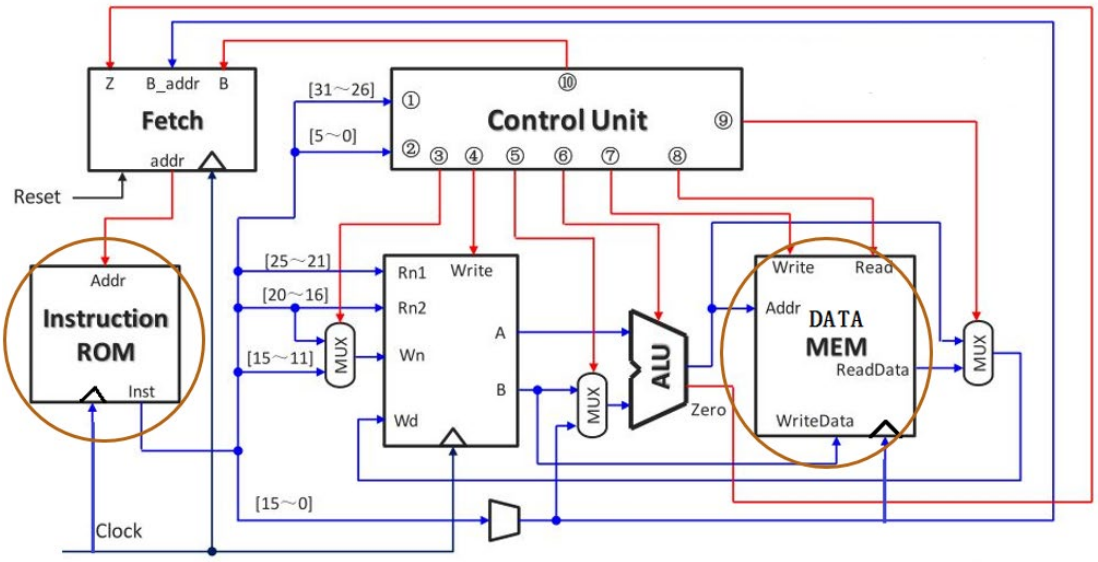
四 实验任务

1. 设计并实现 Main_Board 模块
2. 下载到开发板进行验证

五 实验原理

1. Main_Board 模块的设计

单周期计算机结构如下图所示：



2. 下载到开发板进行验证

六 实验步骤

1. 新建文件：

设置设备和文件路径及编程语言

2. 新建模块：

添加 Verilog Module：

3. 编写程序：

根据上图控制电路功能编写 Verilog 程序

(Main Board)

```
module MainBoard(
    input clk,reset,
    output [31:0] inst,
    output [31:0] PC,
    output [31:0] ALUOUT,
    output [31:0] B_data
    );

    wire [31:0] ADDR_FTI;
    wire [31:0] DATA_DTF;
    wire memwrite, memread;
    wire [31:0] result;
    wire [31:0] nextpc;

    data_flow U0(reset,clk,inst,DATA_DTF,memwrite,memread,result,B_data,nextpc);
    inst_mem U1(nextpc,inst);
    Data_Mem U2(result,memread,memwrite,B_data,clk,DATA_DTF); //Addr,Read,Write,DataIn,Clock,DataOut

    assign PC = nextpc;
    assign ALUOUT = result;

endmodule
```

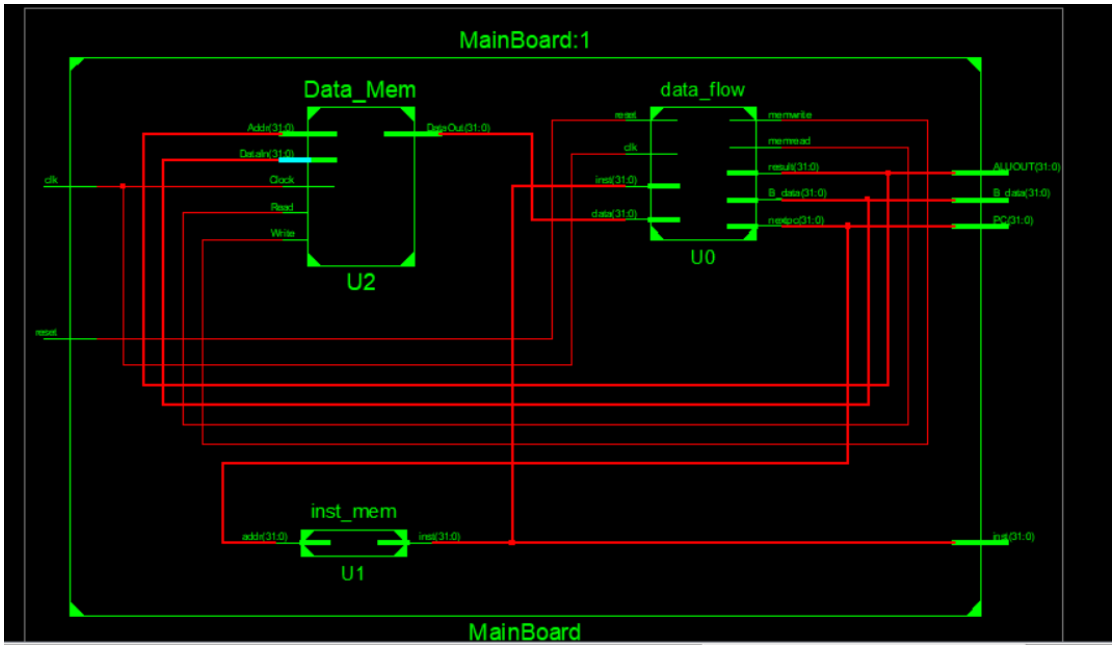
(Top Module)

```
21  module top_module(
22     input Clock,
23     input BTN_IN,
24     input [2:0] SW,
25     input [5:0] LED,
26     input Reset,
27     output wire [7:0] SEGMENT,
28     output wire [5:0] AN
29      );
30
31     reg [23:0] disp_num;
32     wire [2:0] Scanning;
33     wire [31:0] inst;
34     wire [31:0] PC;
35     wire [31:0] ALUOUT;
36     wire [31:0] B_data;
37
38
39     BTN_Anti_Jitter U0(Clock,BTN_IN,BTN_Out);
40     MainBoard U1(BTN_Out,Reset,inst,PC,ALUOUT,B_data);
41
42     always @ (*) begin
43     if (SW[2:0]==3'b000)
44        begin
45        disp_num = inst[23:0];
46        end
```

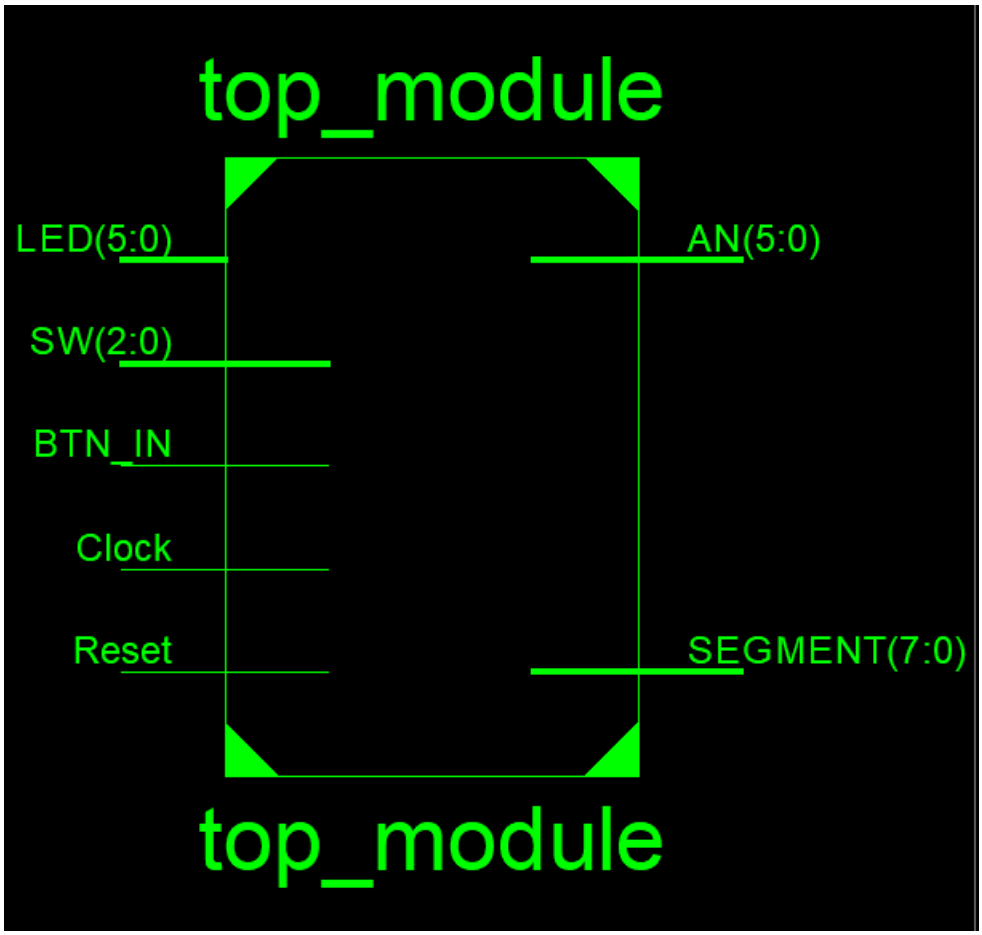
```
60          disp_num = PC[31:8];
61          end
62
63          else if (SW[2:0]==3'b110)
64          begin
65          disp_num = ALUOUT[23:0];
66          end
67
68          else if (SW[2:0]==3'b110)
69          begin
70          disp_num = ALUOUT[31:8];
71          end
72
73          else if (SW[2:0]==3'b111)
74          begin
75          disp_num = B_data[23:0];
76          end
77
78          else if (SW[2:0]==3'b110)
79          begin
80          disp_num = B_data[31:8];
81          end
82      end
83
84      reg [2:0] clockdiv;
85      always @(posedge Clock) clockdiv <= clockdiv + 1;
86      Hex7seg_decode U2(disp_num,clockdiv[2:0],SEGMENT,AN);
87
88  endmodule
89
```

4.综合模块：

(Main Board)



(Top Module)



七 实验结果

实验结果如下：

(Main Board)