实验 6 –CPU 设计-控制器实验报告

姓名：林逸竹　学号：3160104229　专业：计算机科学与技术

课程名称：计算机组成与设计实验　同组学生姓名：无

实验时间：2018-4-16　实验地点：紫金港东 4-509　指导老师：　施青松，黎金洪

一、实验目的和要求

1. 运用寄存器传输控制技术
2. 掌握 CPU 的核心：指令执行过程与控制流关系
3. 设计控制器
4. 学习测试方案的设计
5. 学习测试程序的设计

二、实验内容和原理

2.1 实验任务

1. 设计 9+条指令的控制器

用硬件描述语言设计实现控制器

根据 Exp05 数据通路及指令编码完成控制信号真值表

替换 Exp05 的控制器核

此实验在 Exp05 的基础上完成

2. 设计控制器测试方案

OP 译码测试：R-格式、访存指令、分支指令、转移指令

运算控制测试：Function 译码测试

3. 设计控制器测试程序

2.2 ALU 操作译码

Instruction opcode	ALUop	Instruction operation	Funct field	Desired ALU action	ALU_Control
LW	00	Load word	xxxxxx	Load word	010
SW	00	Store word	xxxxxx	Store word	010
Beq	01	branch equal	xxxxxx	branch equal	110
R-type	1	add	0000	add	010
R-type	1	subtract	0010	subtract	110
R-type	1	AND	0100	AND	000
R-type	1	OR	0101	OR	001
R-type	1	Set on less than	1010	Set on less than	111
R-type	1	NOR	0111	NOR	100

三、主要仪器设备

3.1 实验设备

1. 计算机（Intel Core i5 以上，4GB 内存以上）系统
2. 计算机软硬件课程贯通教学实验系统
3. Xilinx ISE14.4 及以上开发工具

3.2 材料

无

四、实验实现方法、步骤与调试

控制器代码如下：

```
`timescale 1ns / 1ps
`define  CPU_ctrl_signals  {RegDst,  ALUSrc_B,  MemtoReg,  RegWrite,
MemRead, MemWrite, Branch, Jump, ALUop1, ALUop0}
//////////////////////////////////////////////////////////////////////
///////////////
// Company:
// Engineer:
//
// Create Date:    21:12:36 05/26/2014
// Design Name:
// Module Name:    SCPU_ctrl
// Project Name:
// Target Devices:
// Tool versions:
```

```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module SCPU_ctrl( input[5:0]OPcode,                    //OPcode
                  input[5:0]Fun,                       //Function

                   input MIO_ready,                    //CPU Wait

                   output reg RegDst,
                   output reg ALUSrc_B,
                   output reg MemtoReg,
                   output reg Jump,
                   output reg Branch,
                   output reg RegWrite,
                   output mem_w,
                   output reg [2:0]ALU_Control,
                   output CPU_MIO
                 );
reg MemRead, MemWrite;
reg ALUop1, ALUop0;
assign mem_w = MemWrite && (~MemRead);
always@*begin
    case(OPcode)
    6'b000000:begin `CPU_ctrl_signals = 10'b1001000010;end // ALU
    6'b100011:begin `CPU_ctrl_signals = 10'b0111000000;end // load
    6'b101011:begin `CPU_ctrl_signals = 10'b0100010000;end // store
    6'b000100:begin `CPU_ctrl_signals = 10'b0000001001;end // beq
    6'b000010:begin `CPU_ctrl_signals = 10'b0000000110;end // jump
    6'h24:begin    `CPU_ctrl_signals = 10'b0101000011;end // slti
    default:begin   `CPU_ctrl_signals = 10'b0000000000;end
    endcase
end

always@*begin
    case( {ALUop1, ALUop0} )
    2'b00: ALU_Control = 3'b010;              // add
    2'b01: ALU_Control = 3'b110;              // sub
    2'b10:
        case(Fun)
        6'b100000: ALU_Control = 3'b010;     // add
        6'b100010: ALU_Control = 3'b110;      // sub
        6'b100100: ALU_Control = 3'b000;      // and
        6'b100101: ALU_Control = 3'b001;      // or
        6'b101010: ALU_Control = 3'b111;      // slt
        6'b100111: ALU_Control = 3'b100;      // nor
        6'b000010: ALU_Control = 3'b101;      // srl
        6'b010110: ALU_Control = 3'b011;      // xor
        default:   ALU_Control = 3'bx;
        endcase
    2'b11: ALU_Control = 3'b111;                  // slti
    endcase
end
```

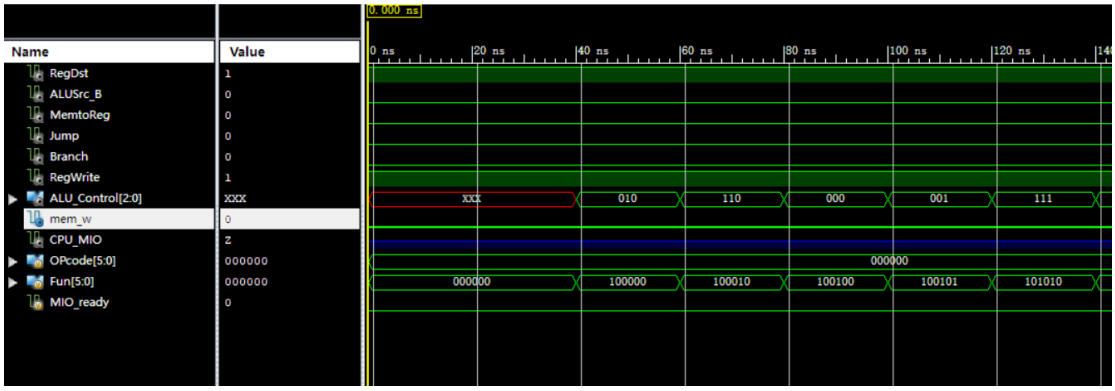
```
endmodule
```

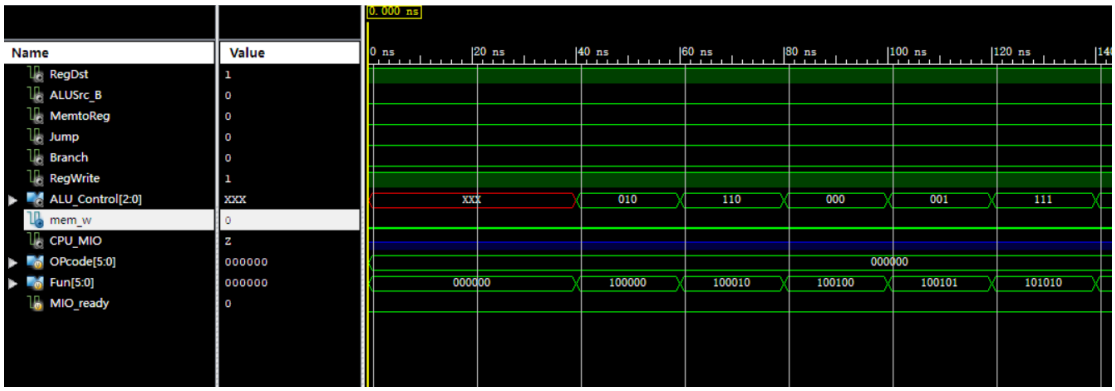
仿真代码如下：

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   20:20:43 04/24/2018
// Design Name:   SCPU_ctrl
// Module Name:   G:/OexpExp04_IP2SCPU/SCPU_ctrl_Test.v
// Project Name:  Oexp02_I0
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: SCPU_ctrl
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module SCPU_ctrl_Test;

    // Inputs
    reg [5:0] OPcode;
    reg [5:0] Fun;
    reg MIO_ready;

    // Outputs
    wire RegDst;
    wire ALUSrc_B;
    wire MemtoReg;
    wire Jump;
    wire Branch;
    wire RegWrite;
    wire [2:0] ALU_Control;
    wire mem_w;
    wire CPU_MIO;

    // Instantiate the Unit Under Test (UUT)
    SCPU_ctrl uut (
        .OPcode(OPcode),
        .Fun(Fun),
        .MIO_ready(MIO_ready),
        .RegDst(RegDst),
        .ALUSrc_B(ALUSrc_B),
        .MemtoReg(MemtoReg),
        .Jump(Jump),
        .Branch(Branch),
        .RegWrite(RegWrite),
        .ALU_Control(ALU_Control),
```

```
        .mem_w(mem_w),
        .CPU_MIO(CPU_MIO)
    );

    initial begin
        // Initialize Inputs
        OPcode = 0;
        Fun = 0;
        MIO_ready = 0;
        #40;
        Fun = 6'b100000;  // add
        #20;
        Fun = 6'b100010;  // sub
        #20;
        Fun = 6'b100100;  // and
        #20;
        Fun = 6'b100101;  // or
        #20;
        Fun = 6'b101010;  // slt
        #20;
        Fun = 6'b100111;  // nor
        #20;
        Fun = 6'b000010;  // srl
        #20;
        Fun = 6'b010110;  // xor
        #20;
        Fun = 6'b111111;  // 间隔
        #1;
        OPcode = 6'b100011;   // load
        #20;
        OPcode = 6'b101011;   // store
        #20;
        OPcode = 6'b000100;   // beq
        #20;
        OPcode = 6'b000010;   // jump
        #20;
        OPcode = 6'h24;       // slti
        #20;
        OPcode = 6'h3f;
        Fun = 6'b0000000;

    end

endmodule
```

仿真结果如下:





五、实验结果与分析

能够实现下表功能。

图形功能测试

开关	位置	功能
SW[1:0]	X0	七段码图形显示
SW[2]	0	CPU全速时钟
SW[4:3]	00	7段码从上至下亮点循环右移
SW[4:3]	11	7段码矩形从下到大循环显示
SW[7:5]	000	作为外设使用（E0000000/FFFFFE00）

文本功能测试

开关	位置	功能
SW[1:0]	01	七段码文本显示（低16位） Arduino有效
	11	七段码文本显示（高16位）
SW[2]	0	CPU全速时钟
SW[4:3]	01	7段码显示RAM数字
SW[4:3]	10	7段码显示累加
SW[7:5]	000	作为外设使用（E0000000/FFFFFE00）

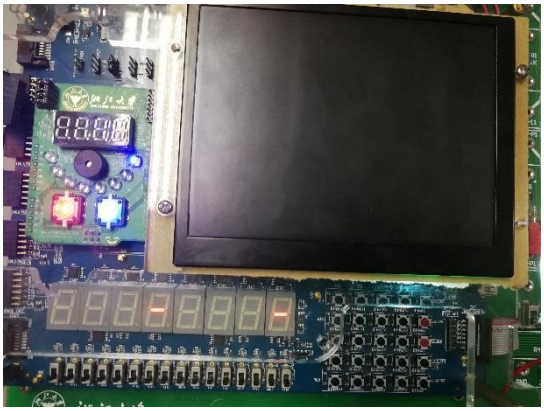


Figure 1 实验结果(1) 跑马灯

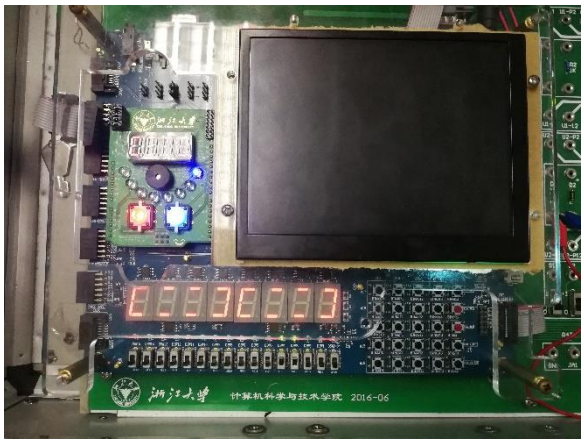


Figure 2 实验结果(2) 矩形

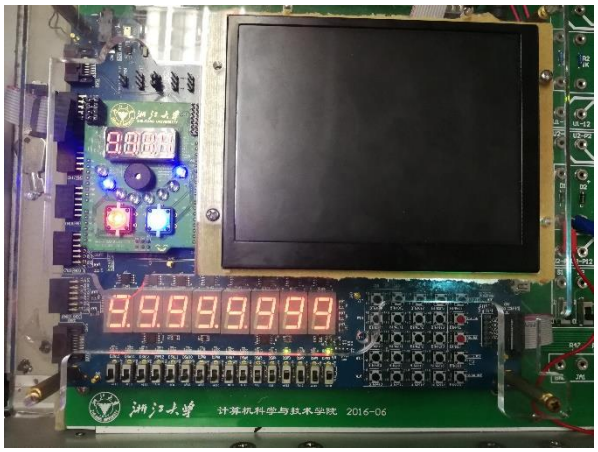


Figure 3 实验结果(3) RAM

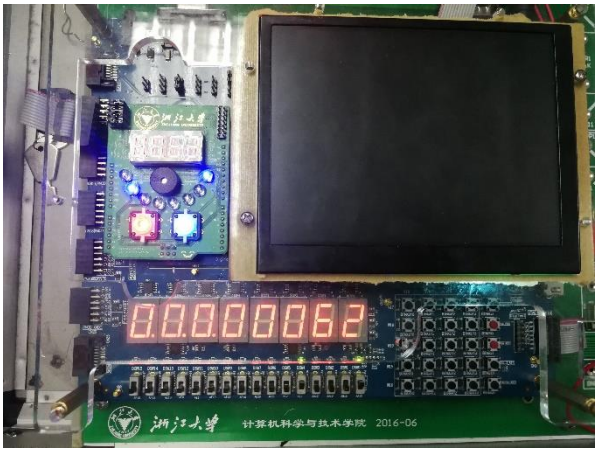


Figure 4 实验结果(4) 累加

六、讨论、心得

6.1 思考题

6.1.1 单周期控制器时序体现在哪里?

答：每个时钟执行一条 PC 指令

6.1.2 设计 bne 指令需要增加控制信号吗？

答：可以利用 ALU 的 zero 信号来判断，如果要增加控制信号的话也可以，需要增添一个信号和 zero 的反作并操作，再和 branch 指令进行或操作。

6.1.3 扩展下列指令，控制器将作如何修改。

答：见实验 7.

6.1.4 动态存储模块测试七段显示会出现什么问题？

答：会出现时序上的问题。

6.2 心得

控制器使用 verilog 语言来实现会方便得多，后续进行指令的增添要容易，而且更能与实际理解上产生对应，以后要尽量都用 verilog 语言来实现。