

lab6 CPU综合设计

曾鄧琛 PB20071431 物理学院辅计

一、实验题目

CPU综合设计

二、实验目的：

1. 理解计算机硬件系统的组成结构和工作原理
2. 掌握软硬件综合系统的设计和调试方法
3. 进一步理解 CPU 的结构和工作原理，熟练掌握数据通路和控制器的设计和描述方法
4. 在lab5基础上扩充指令集，设计应用程序，完成排序算法

三、实验平台：

Vivado

四、实验过程展示

1.控制器模块

```
module control(input en,
               input [31:0] Instruction,
               output reg_base,
               output jal,
               output Branch,
               output [1:0] RegSrc,
               output [3:0] alu_control,
               output MemWrite,
               output MemRead,
               output ALUSrc1,
               output ALUSrc2,
               output RegWrite,
               output [1:0] readXF,
               output writeXF,
               output [2:0] fcontrol);

    wire [1:0] ALUop;
    wire [3:0] alucontrol;
    wire [6:0] opcode = Instruction[6:0];
    assign reg_base = en === 1'b0 ? 1'b0 : (opcode == 7'b1100111);
    assign jal = en === 1'b0 ? 1'b0 :
        (opcode[2] && opcode[6:5] == 2'b11);
    assign Branch = en === 1'b0 ? 1'b0 :
        (opcode == 7'b1100011);
    assign ALUSrc1 = en === 1'b0 ? 1'b0 :
        (opcode == 7'b0010111);
    assign ALUSrc2 = en === 1'b0 ? 1'b0 :
        (opcode != 7'b0110011 && opcode != 7'b1010011 && opcode != 7'b1100011);
    assign ALUop = en === 1'b0 ? 1'b0 :
        {opcode[5] & opcode[4], opcode[6] & opcode[5]};
    ALU_control ALUControl(
```

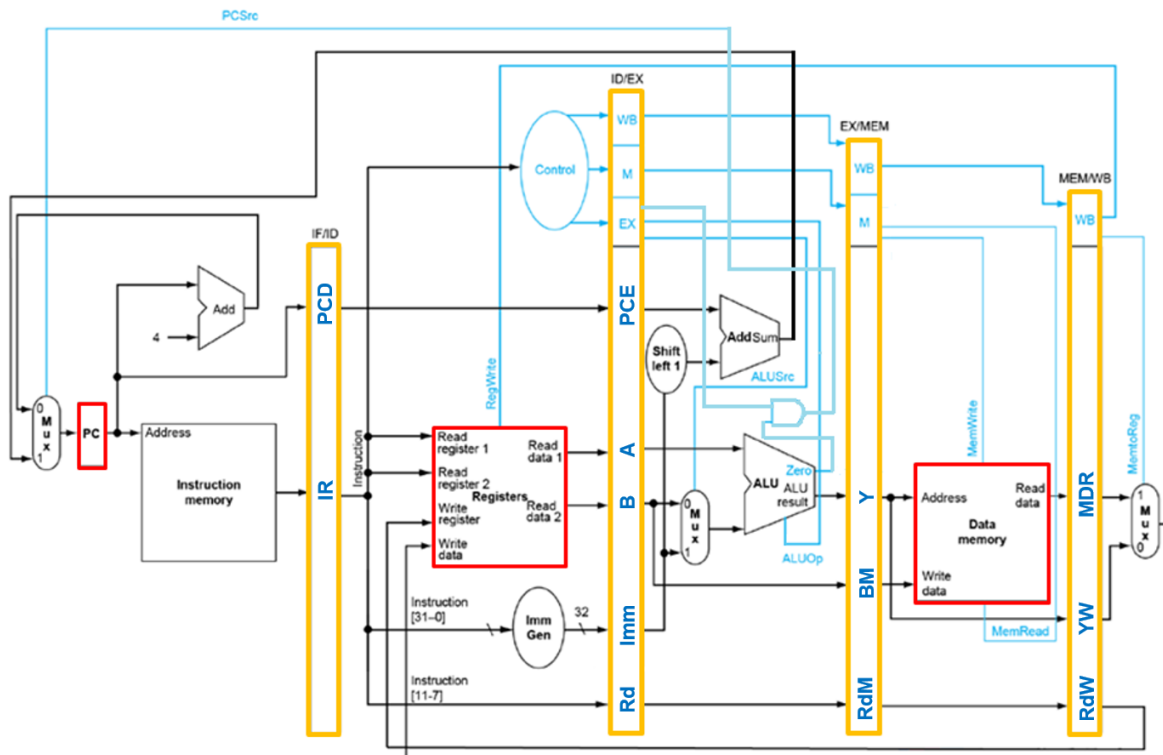
```

// .ALUop(ALUop),
.Instruction({Instruction[4:0] == 5'b10111, Instruction[25], ALUop,
Instruction[30], Instruction[14:12]}),
.control(alucontrol)
);
assign alu_control = en == 1'b0 ? 4'b0000 : alucontrol;

assign MemWrite = en == 1'b0 ? 1'b0 : (opcode == 7'b0100011 || opcode ==
7'b0100111);
assign MemRead = en == 1'b0 ? 1'b0 : ((opcode == 7'b0000011 || opcode ==
7'b0000111) ? 1'b1 : 1'b0);

assign RegWrite = en == 1'b0 ? 1'b0 : (opcode != 7'b0100011 && opcode !=
7'b0100111 && opcode != 7'b1100011);
assign RegSrc = en == 1'b0 ? 2'b00 : (opcode[4] ? 2'b00 : ((opcode[6] ==
0) ? 2'b01 : 2'b10));
assign readXF = en == 1'b0 ? 2'b00 : (opcode == 7'b0100111 ? 2'b10 :
(opcode == 7'b1010011 ? 2'b11 : 2'b00));
assign writeXF = en == 1'b0 ? 1'b0 : (opcode == 7'b0000111);
assign fcontrol = en == 1'b0 ? 3'b111: Instruction[14:12];
endmodule

```



控制器与lab5中相比增设了readXF、writeXF、fcontrol，为了在排序算法中解决问题。

2.ALU模块

```

module ALU_control(input [7:0] Instruction,
                  output [3:0] control);
    reg [3:0] temp;
    always @(*) begin
        temp = 4'b1000;
        if (Instruction[7:0] == 8'b0110_0000) begin
            // 4: mul
            temp = 4'b0100;
        end
    end
endmodule

```

```

end
else if (Instruction[7:0] == 8'b0110_0100) begin
    // 5: div
    temp = 4'b0101;
end
else if (Instruction[7:0] == 8'b0010_1000 || Instruction[5:4] == 2'b01
|| (Instruction[2:0] == 3'b000 && Instruction[4] == 1'b1)) begin
    // 6: sub
    temp = 4'b0110;
end
else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b001) begin
    // 7: sll
    temp = 4'b0111;
end
else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b101) begin
    // 8: srl
    temp = 4'b1000;
end
else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b100 &&
(Instruction[6] == 1'b0 || Instruction[5] == 1'b0)) begin
    // 3: xor
    temp = 4'b0011;
end
else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b110) begin
    // 1: or
    temp = 4'b0001;
end
else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b111) begin
    // 0: and
    temp = 4'b0000;
end
else begin
    // 2: add
    temp = 4'b0010;
end
end
assign control = temp;
endmodule

```

与上一次实验相比增设了or、xor、srl、sll、sub等指令集，对ALU进行重新编写，并更改control中的instruction。

```

module ALU_control(input [7:0] Instruction,
                   output [3:0] control);
    reg [3:0] temp;
    always @(*) begin
        temp = 4'b1000;
        if (Instruction[7:0] == 8'b0110_0000) begin
            // 4: mul
            temp = 4'b0100;
        end
        else if (Instruction[7:0] == 8'b0110_0100) begin
            // 5: div
            temp = 4'b0101;
        end
        else if (Instruction[7:0] == 8'b0010_1000 || Instruction[5:4] == 2'b01
|| (Instruction[2:0] == 3'b000 && Instruction[4] == 1'b1)) begin

```

```

        // 6: sub
        temp = 4'b0110;
    end
    else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b001) begin
        // 7: sll
        temp = 4'b0111;
    end
    else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b101) begin
        // 8: srl
        temp = 4'b1000;
    end
    else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b100 &&
(Instruction[6] == 1'b0 || Instruction[5] == 1'b0)) begin
        // 3: xor
        temp = 4'b0011;
    end
    else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b110) begin
        // 1: or
        temp = 4'b0001;
    end
    else if (Instruction[7] == 1'b0 && Instruction[2:0] == 3'b111) begin
        // 0: and
        temp = 4'b0000;
    end
    else begin
        // 2: add
        temp = 4'b0010;
    end
end
end
assign control = temp;
endmodule

```

3.应用程序：冒泡排序算法

```

module Compare( input [31:0] rs1,
                input [31:0] rs2,
                input [2:0] control,
                output [31:0] result
                );
    reg [31:0] temp;
    reg eqresult;
    reg ltresult;
    reg signed [7:0] exp1;
    reg signed [7:0] exp2;

    always @(*) begin
        eqresult = (rs1 == rs2);
        if (rs1[31] != rs2[31]) begin
            ltresult = (rs1[31] == 1'b1);
        end
        else begin
            exp1 = rs1[30:23] - 127;
            exp2 = rs2[30:23] - 127;
            if (exp1 < exp2) begin
                ltresult = 1'b1;
            end
            else if (exp1 == exp2) begin

```

```

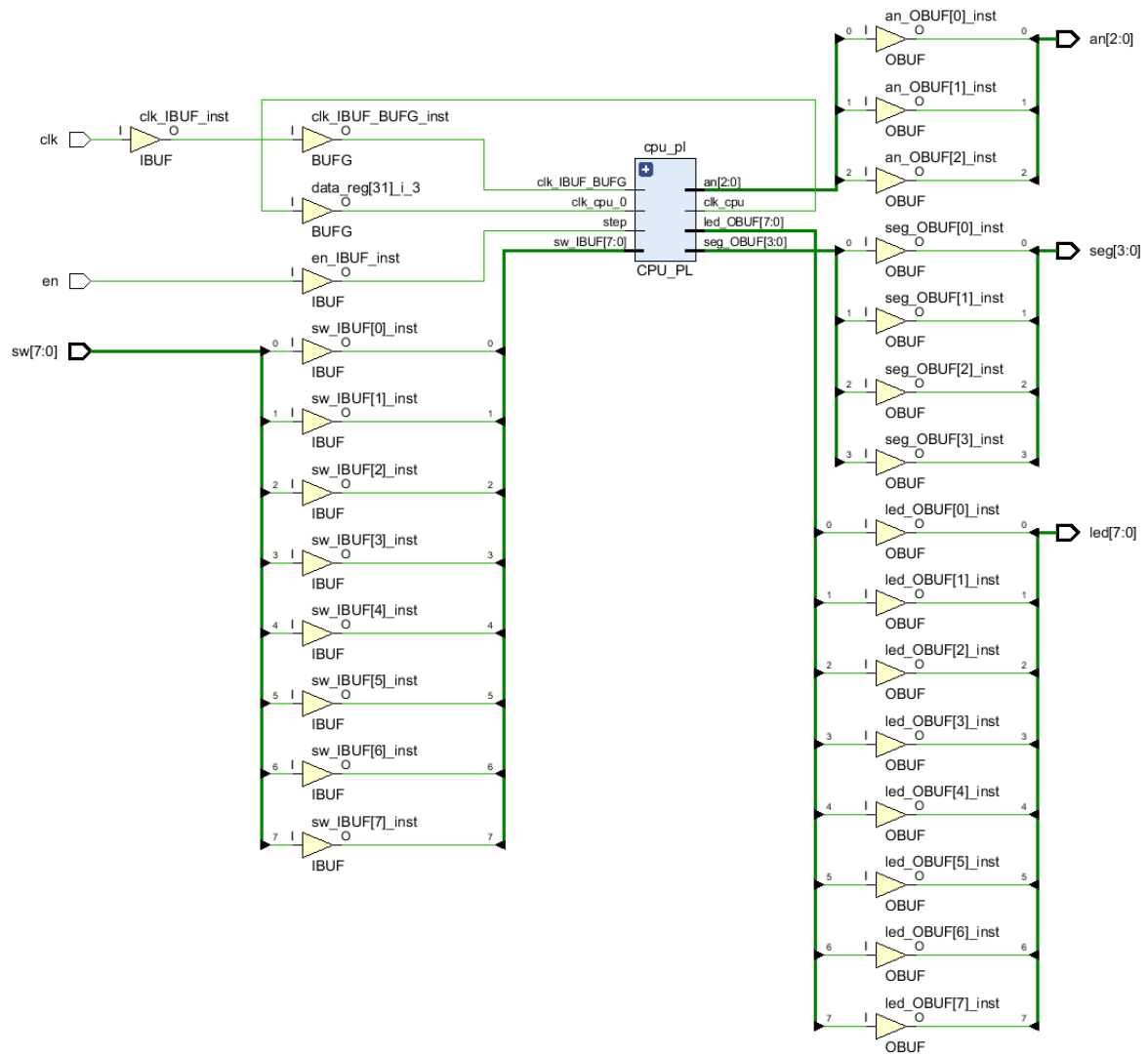
        ltresult = (rs1[22:0] < rs2[22:0]);
    end
    else begin
        ltresult = 1'b0;
    end
    ltresult = rs1[31] == 1'b0 ? ltresult : ~ltresult;
end
end

always @(*) begin
    case(control)
        // 0: le
        3'b000: temp = (eqresult | ltresult);
        // 1: lt
        3'b001: temp = ltresult;
        // 2: eq
        3'b010: temp = eqresult;
        // 3: gt
        3'b011: temp = !(eqresult & ltresult);
        // 4: neq
        3'b100: temp = !eqresult;
        // 5: ge
        3'b101: temp = !ltresult;
        default: temp = 1'b0;
    endcase
end
assign result = {31'b0, temp};
endmodule

```

按照RISC v指令集手册，对照利用取exp的方式对float数比大小，从而进行冒泡排序。rs数据扩展到32位字节，采用 $\text{exp1} = \text{rs1}[30:23] - 127$ ，首先比较exp位，再次比较rs1[22:0]，选择出小者作为结果。此后设计一个时序逻辑，对是否相等、大或小进行判断，从而选择跳转程序。

4.TOP文件



此处仅展示重要的部分内容：

```

assign io_addr = y[7:0];
assign io_we    = y[10] & mm[1];
assign io_dout = (y[10] & mm[1]) ? bm : 0;
// control signal
control Control(
    .en(control_signal),
    .Instruction(ir),
    .reg_base(reg_base),
    .jal(jal),
    .Branch(Branch),
    .RegSrc(RegSrc),
    .alu_control(alu_control),
    .MemWrite(MemWrite),
    .MemRead(MemRead),
    .ALUSrc1(ALUSrc1),
    .ALUSrc2(ALUSrc2),
    .RegWrite(RegWrite),
    .readXF(readXF),
    .writeXF(writeXF),
    .fcontrol(fcontrol)
);
// ID/EX
segment_register PCE(.en(1'b1), .clk(clk), .flush(1'b0), .in(pcd),
    .out(pce));

```

```

    segment_register A(.en(1'b1), .clk(clk), .flush(1'b0), .in(reg_read_data1),
.out(a));
    segment_register B(.en(1'b1), .clk(clk), .flush(1'b0), .in(reg_read_data2),
.out(b));
    segment_register Imm(.en(1'b1), .clk(clk), .flush(1'b0), .in(immediate),
.out(imm));
    segment_register #(.WIDTH(5)) Rd(.en(1'b1), .clk(clk), .flush(1'b0),
.in(ir[11:7]), .out(rd));
    segment_register #(.WIDTH(5)) Rs1(.en(1'b1), .clk(clk), .flush(1'b0),
.in(ir[19:15]), .out(rs1));
    segment_register #(.WIDTH(5)) Rs2(.en(1'b1), .clk(clk), .flush(1'b0),
.in(ir[24:20]), .out(rs2));
    // control
    segment_register #(.WIDTH(14)) EX(.en(1'b1), .clk(clk), .flush(flush),
.in({readXF, fcontrol, reg_base, jal, Branch, ALUSrc2, ALUSrc1, alu_control}),
.out(ex));
    segment_register #(.WIDTH(2)) M(.en(1'b1), .clk(clk), .flush(flush),
.in({MemWrite, MemRead}), .out(m));
    segment_register #(.WIDTH(4)) WB(.en(1'b1), .clk(clk), .flush(flush),
.in({writeXF, RegWrite, RegSrc}), .out(wb));
    segment_register #(.WIDTH(3)) J(.en(1'b1), .clk(clk), .flush(flush),
.in(ir[14:12]), .out(j_control));

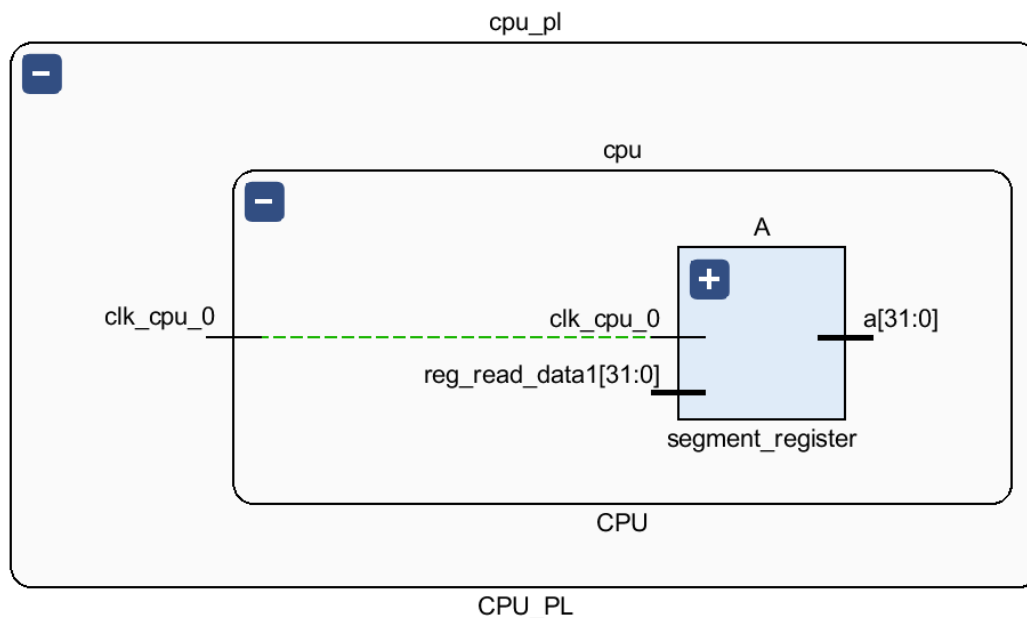
```

```

    // EX/MEM
    segment_register PCM(.en(1'b1), .clk(clk), .flush(1'b0), .in(pce),
.out(pcm));
    segment_register Y(.en(1'b1), .clk(clk), .flush(1'b0),
.in(calculation_result), .out(y));
    segment_register BM(.en(1'b1), .clk(clk), .flush(1'b0), .in(b1), .out(bm));
    segment_register #(.WIDTH(5)) RdM(.en(1'b1), .clk(clk), .flush(1'b0),
.in(rd), .out(rdm));
    // control
    segment_register #(.WIDTH(2)) MM(.en(1'b1), .clk(clk), .flush(1'b0), .in(m),
.out(mm));
    segment_register #(.WIDTH(4)) WBM(.en(1'b1), .clk(clk), .flush(1'b0),
.in(wb), .out(wbm));

    segment_register CTRLM(.en(1'b1), .clk(clk), .flush(1'b0), .in(ctrl),
.out(ctrlm));

```



五、FPGA与.coe文件对应检查

1.数列的冒泡排序

FPGA interface

led7 led6 led5 led4 led3 led2 led1 led0

G18 F18 E17 D17 G17 E18 D18 C17

FPGA
XC7A100T-CSG324-1

H16 G13 F13 E16 H14 G16 F16 D14

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

uart show>

```
FPGAOL UART xterm.js 1.1
```

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

input

segplay(sharing with led) hexplay

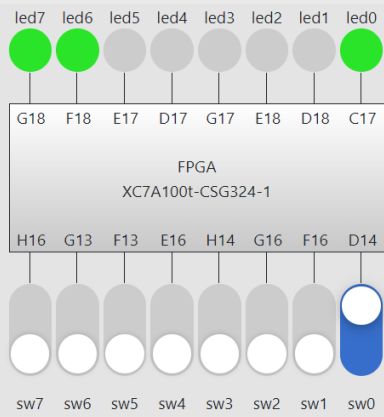
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

FPGA interface



uart show>

FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

input

segplay(sharing with led) hexplay



segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock

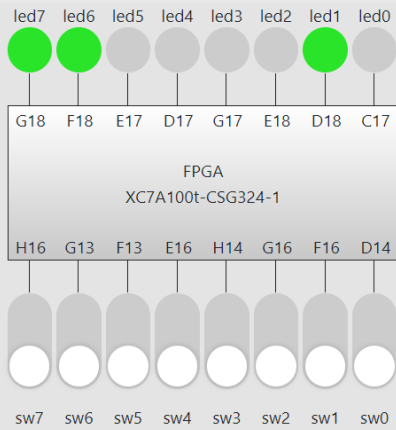
button

None



clk btn pins: clk_btn
xdc,ucf sym: B18

FPGA interface



uart show>

FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

input

segplay(sharing with led) hexplay



segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock

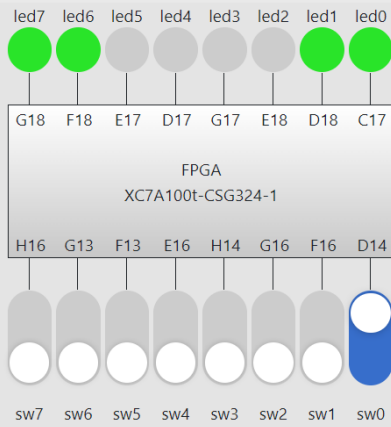
button

None



clk btn pins: clk_btn
xdc,ucf sym: B18

FPGA interface



uart show>

FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

input

segplay(sharing with led)

hexplay



segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

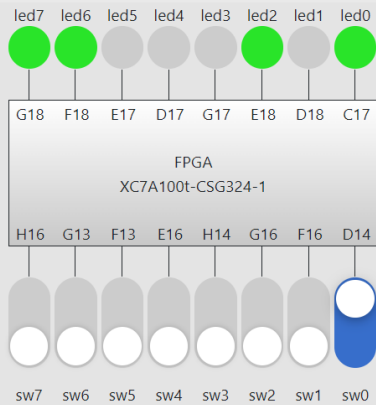
soft clock

button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

FPGA interface



uart show>

FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

input

segplay(sharing with led)

hexplay



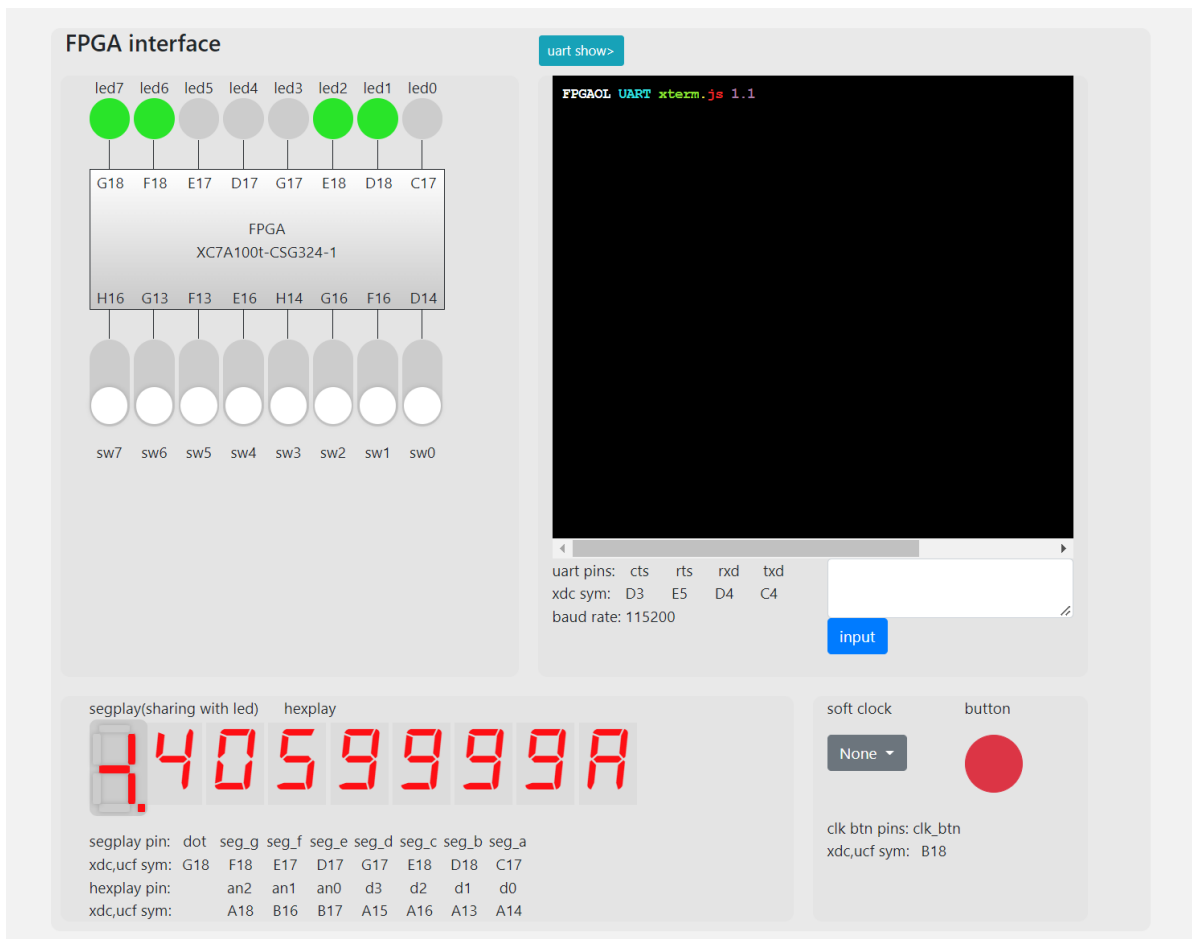
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock

button

None

clk btn pins: clk_btn
xdc,ucf sym: B18



以下为冒泡排序主要汇编代码：结果显示与FPGA上板结果相同。

```
li    t0, 0
loop1:
li     t1, 0
li     t5, 0
loop2:
# if (arr[j] > arr[j + 1])
slli   t6, t1, 2
add     t2, t6, a0
# lw    t3, 0(t2)
flw     f3, 0(t2)
# lw    t4, 4(t2)
flw     f4, 4(t2)
# ble   t3, t4, continue
fle.s   t3, f4, f3
beqz    t3, continue
# swap
# sw    t3, 4(t2)
fsw     f3, 4(t2)
# sw    t4, 0(t2)
fsw     f4, 0(t2)
# set flag
li      t5, 1

continue:
# j++
addi    t1, t1, 1
# if (j < len - 1 - i)
addi    t6, t0, 1
sub     t6, a1, t6
```

```

blt      t1, t6, loop2

beqz     t5, finishSort
# i++
addi     t0, t0, 1
# if (i < len - 1)
addi     t6, a1, -1
blt      t0, t6, loop1
finishSort:
ret

```

2.单步控制分支检验

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00003000	0xffffd517	auipc x10,0xffffffff	9: la a0, array
<input type="checkbox"/>	0x00003004	0x00050513	addi x10,x10,0	
<input type="checkbox"/>	0x00003008	0xffffd597	auipc x11,0xffffffff	10: lw a1, length
<input type="checkbox"/>	0x0000300c	0x0205a583	lw x11,0x00000020(x...	
<input type="checkbox"/>	0x00003010	0x084000ef	jal x1,0x00000084	11: jal printArray
<input type="checkbox"/>	0x00003014	0xffffd517	auipc x10,0xffffffff	13: la a0, array
<input type="checkbox"/>	0x00003018	0xfec50513	addi x10,x10,0xffff...	
<input type="checkbox"/>	0x0000301c	0xffffd597	auipc x11,0xffffffff	14: lw a1, length
<input type="checkbox"/>	0x00003020	0x00c5a583	lw x11,12(x11)	
<input type="checkbox"/>	0x00003024	0x01c000ef	jal x1,0x0000001c	15: jal bubbleSort
<input type="checkbox"/>	0x00003028	0xffffd517	auipc x10,0xffffffff	17: la a0, array
<input type="checkbox"/>	0x0000302c	0xfd850513	addi x10,x10,0xffff...	
<input type="checkbox"/>	0x00003030	0xffffd597	auipc x11,0xffffffff	18: lw a1, length
<input type="checkbox"/>	0x00003034	0xff85a583	lw x11,0xffffffff(x...	

FPGA interface

led7 led6 led5 led4 led3 led2 led1 led0

G18 F18 E17 D17 G17 E18 D18 C17

FPGA
XC7A100T-CSG324-1

H16 G13 F13 E16 H14 G16 F16 D14

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

uart show>

```
FPGAOL UART xterm.js 1.1
```

uart pins: cts rts rxd txd

xdc sym: D3 E5 D4 C4

baud rate: 115200

input

segplay(sharing with led) hexplay

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a

xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17

hexplay pin: an2 an1 an0 d3 d2 d1 d0

xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

soft clock button

None

clk btn pins: clk_btn

xdc,ucf sym: B18

两次en后结果对应一致，auipc验证完毕。

<input type="checkbox"/>	0x00003090	0x00008067	jalr x0,x1,0	73: ret
<input type="checkbox"/>	0x00003094	0x00050293	addi x5,x10,0	78: addi t0, a0, 0
<input type="checkbox"/>	0x00003098	0xffff58593	addi x11,x11,0xffff...	80: addi a1, a1, -1
<input type="checkbox"/>	0x0000309c	0x0005ca63	blt x11,x0,0x00000014	81: bltz a1, finishPrint
<input type="checkbox"/>	0x000030a0	0x0002a503	lw x10,0(x5)	82: lw a0, 0(t0)
<input type="checkbox"/>	0x000030a4	0x40a02423	sw x10,0x00000408(x0)	85: sw a0, 0x408(x0)
<input type="checkbox"/>	0x000030a8	0x00428293	addi x5,x5,4	86: addi t0, t0, 4

FPGA interface

led7 led6 led5 led4 led3 led2 led1 led0

G18 F18 E17 D17 G17 E18 D18 C17

FPGA
XC7A100t-CSG324-1

H16 G13 F13 E16 H14 G16 F16 D14

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

uart show>

FPGAOL UART xterm.js 1.1

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

input

segplay(sharing with led) hexplay

400003094









segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

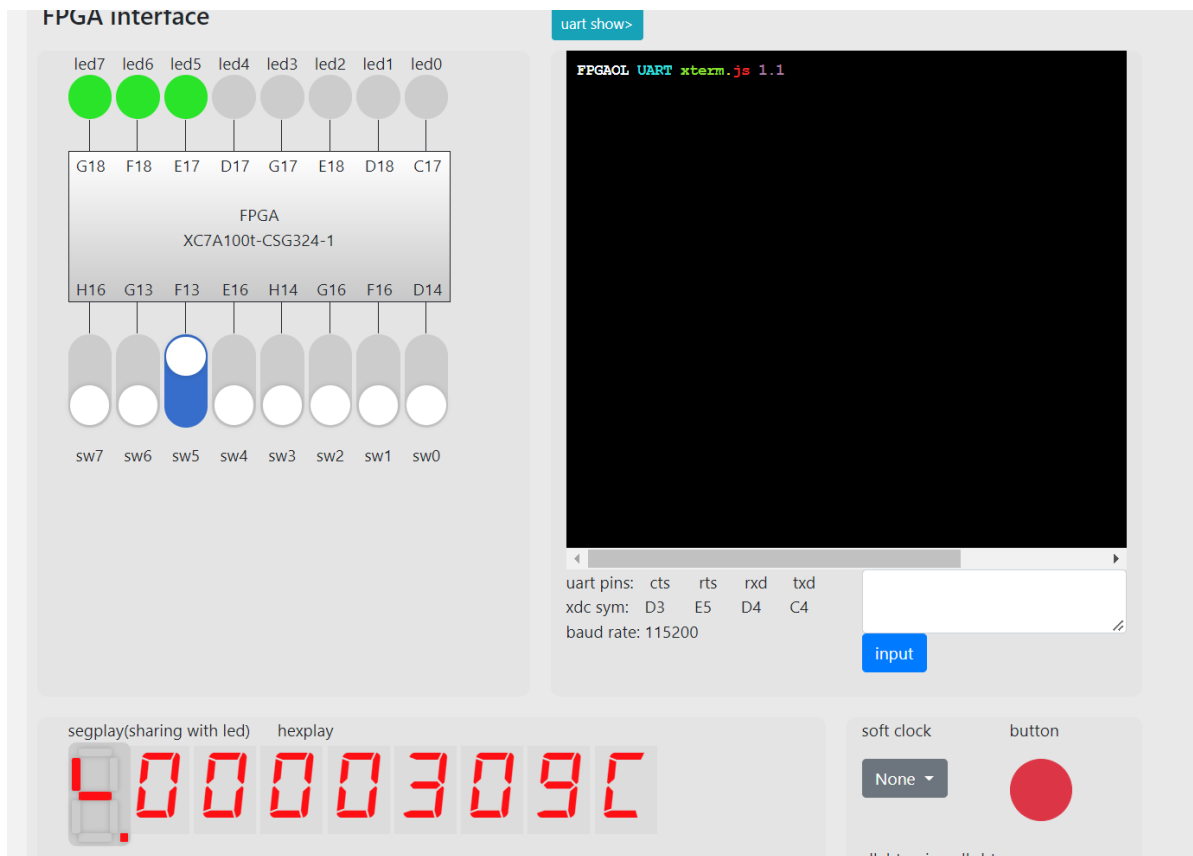
soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

此过程两者en次数不同，在于jalr的跳转在assembly里面直接跳转，而在fpga上需要en才跳转成功，故而相差2次en。

	0x00003094	0x00050293	addi x5,x10,0	78:	addi	t0, a0, 0
	0x00003098	0xffff58593	addi x11,x11,0xffff..	80:	addi	a1, a1, -1
	0x0000309c	0x0005ca63	blt x11,x0,0x00000014	81:	bltz	a1, finishPrint
	0x000030a0	0x0002a503	lw x10,0(x5)	82:	lw	a0, 0(t0)
	0x000030a4	0x40a02423	sw x10,0x00000408(x0)	85:	sw	a0, 0x408(x0)
	0x000030a8	0x00428293	addi x5,x5,4	86:	addi	t0, t0, 4
	0x000030ac	0xfedff06f	jal x0,0xfffffffec	87:	j	loop
	0x000030b0	0x00008067	jalr x0,x1,0	89:	ret	



bltz验证如上。

其他例如lw.jal.slli, flw fsw(浮点数) beqz addi sub blt jalr等操作在检查时——演示过了，篇幅有限此处不——展示。

六、总结与展望

本学期计组实验到此告一段落，从开始的简单寄存器堆读写到单周期、流水线.....硬件代码一路写下来也挺难受的，特别对于辅修计算机的物院同学而言，时间就更不够了，但这门课对于物电专业还很有用处的，之后芯片设计与高性能计算优化都需要用到。总之，完结撒花🎉~体系结构再见~~~