

Lab2

1. 控制器和立即数模块代码

```
lab2 > src >  sc_cu.v
 20  module sc_cu (inst, z, wmem, wreg, m2reg, aluc,
 19  |   |   |   aluimm, pcsource, sext,i_lui,i_sw,shift);
 18  input  [31:0] inst;
 17  input      z;
 16  output     wreg,m2reg,aluimm,sext,wmem,i_lui,i_sw,shift;
 15  output  [3:0] aluc;
 14  output  [1:0] pcsource;
 13  wire [6:0] op = inst[6:0];
 12  wire [2:0] func3 = inst[14:12];
 11  wire [6:0] func7 = inst[31:25];
 10
  9  wire r_type = (op == 7'b0110011);
  8  wire i_type = (op == 7'b0010011);
  7 //R type
  6  wire i_add = r_type & (func3 == 3'b000 ) & ~inst[30];           //000, 0
  5  wire i_sub = r_type & (func3 == 3'b000 )& inst[30];             //000, 1
  4
  3
  2
  1 // please complete the deleted code.
 21
  1
  2  wire i_and = r_type & (func3 == 3'b111 );                  //111
  3  wire i_or  = r_type & (func3 == 3'b110 );                  //110
  4  wire i_xor = r_type & (func3 == 3'b100 );                  //100
  5  wire i_sll = r_type & (func3 == 3'b001 );                  //001
  6  wire i_srl = r_type & (func3 == 3'b101 ) & ~inst[30];       //101, 0
  7  wire i_sra = r_type & (func3 == 3'b101 ) & inst[30] ;       //101,1
  8
  9 //I type
 10  wire i_addi = i_type & (func3 == 3'b000) ;                 //000
 11  wire i_andi = i_type & (func3 == 3'b111) ;                 //111
 12  wire i_ori = i_type & (func3 == 3'b110) ;                 //110
 13  wire i_xori = i_type & (func3 == 3'b100) ;                //100
 14  wire i_slli = i_type & (func3 == 3'b001) ;                //001
 15  wire i_srli = i_type & (func3 == 3'b101 ) & ~inst[30];     //101, 0
 16  wire i_srai = i_type & (func3 == 3'b101 ) & inst[30] ;     //101,1
 17
 18  wire i_lw   = (op == 7'b0000011) & (func3 == 3'b010) ; //010
 19  wire i_jalr = (op == 7'b1100111) & (func3 == 3'b000) ; //000
 20
 21 //S type
 22  wire i_sw   = (op == 7'b0100011) & (func3 == 3'b010) ; //010
 23 //SB type
```

图1: 控制器模块代码-第一部分

根据各指令在RISC-V-Reference-Data中的格式定义，补充I-type, S-type指令的判断逻辑

```

23 //SB type
24 wire i_beq = (op == 7'b1100011) & (func3 == 3'b000) ; //000
25 wire i_bne = (op == 7'b1100011) & (func3 == 3'b001) ; //001
26 //U type
27 wire i_lui = (op == 7'b0110111) ;
28 //UJ type
29 wire i_jal = (op == 7'b1101111) ;
30
31 assign pcsource[1] = i_jalr | i_jal;
32 assign pcsource[0] = ( i_beq & z ) | (i_bne & ~z) | i_jal ;
33
34 assign wreg = i_add | i_sub | i_and | i_or | i_xor |
35 | | | | i_sll | i_srl | i_sra | i_addi | i_andi |
36 | | | | i_ori | i_xori | i_srli | i_slli | i_srai | i_lw | i_jalr | i_lui | i_jal;
37
38
39 assign aluc[3] = i_sub | i_sra | i_srai | i_beq | i_bne;
40 assign aluc[2] = i_and | i_or | i_xor | i_srl | i_sra | i_andi | i_ori | i_xori | i_srli | i_srai;
41 assign aluc[1] = i_and | i_or | i_andi | i_ori | i_lui;
42 assign aluc[0] = i_and | i_sll | i_srl | i_sra | i_andi | i_slli | i_srli | i_srai;
43
44 assign aluimm = i_addi | i_andi | i_ori | i_xori | i_slli | i_srli | i_srai | i_lw | i_sw | i_jalr | i_lui;
45 assign sext = i_addi | i_lw | i_jalr | i_sw | i_beq | i_bne | i_jal | i_type; //add itype
46 assign wmem = i_sw;
47 assign m2reg = i_lw;
48 assign shift = i_srli | i_slli | i_srai | i_sll | i_srl | i_sra;;
49 endmodule

```

图2: 控制器模块代码-第二部分

根据各指令产生信号的情况，补充产生控制信号的逻辑。

```

lab2 > src > immext.v
10 module immext(inst, pcsource, sext, i_lui, i_sw, shift, out_immediate);
  9   input [31:0] inst;
  8   input sext, i_lui, i_sw, shift;
  7   input [1:0] pcsource;
  6   output [31:0] out_immediate;
  5   wire e = sext & inst[31];
  4   wire [31:0] lui_imm = {inst[31:12], 12'b0};
  3   wire [31:0] shift_imm = {{27{1'b0}}, inst[24:20]};
  2   wire [31:0] sw_imm = {{20{e}}, inst[31:25], inst[11:7]};
  1   wire [31:0] branchpc_offset = {{19{e}}, inst[31], inst[7], inst[30:25], inst[11:8], 1'b0};
  11  wire [31:0] jalpc_offset = {{11{e}}, inst[31], inst[19:12], inst[20], inst[30:21], 1'b0};
  1   wire [31:0] itype_imm = {{20{e}}, inst[31:20]};
  2   assign out_immediate =
  3     i_lui ? lui_imm :
  4     i_sw ? sw_imm :
  5     shift ? shift_imm :
  6     (pcsource == 2'b01) ? branchpc_offset :
  7     (pcsource == 2'b11) ? jalpc_offset :
  8     itype_imm;
  9 endmodule

```

图3: 立即数模块代码

先定义各个指令中立即数的产生逻辑，再根据实际接收到的指令信号用assign语句组合逻辑产生立即数。

2. Vivado工程文件树截图

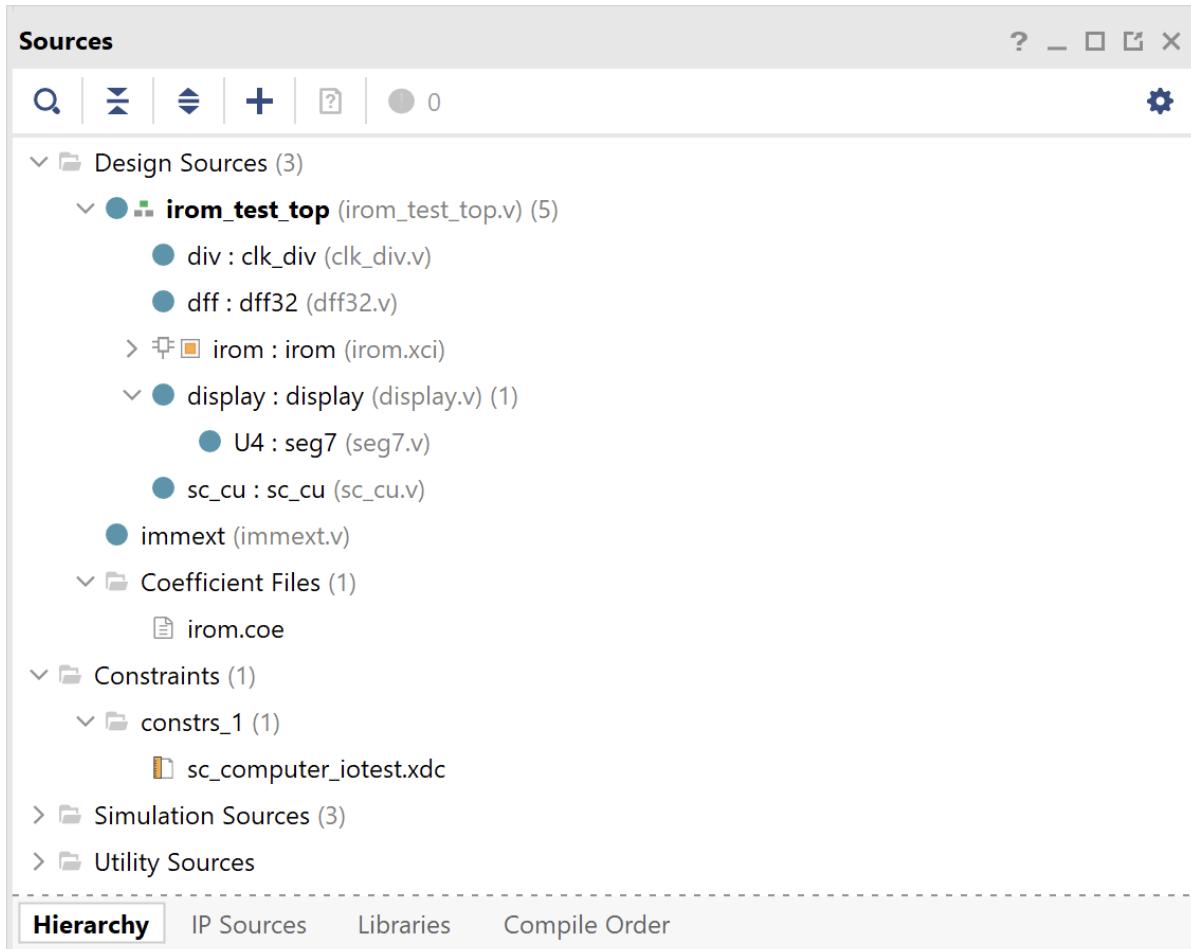


图4: Vivado工程文件树截图

3. 板级测试照片

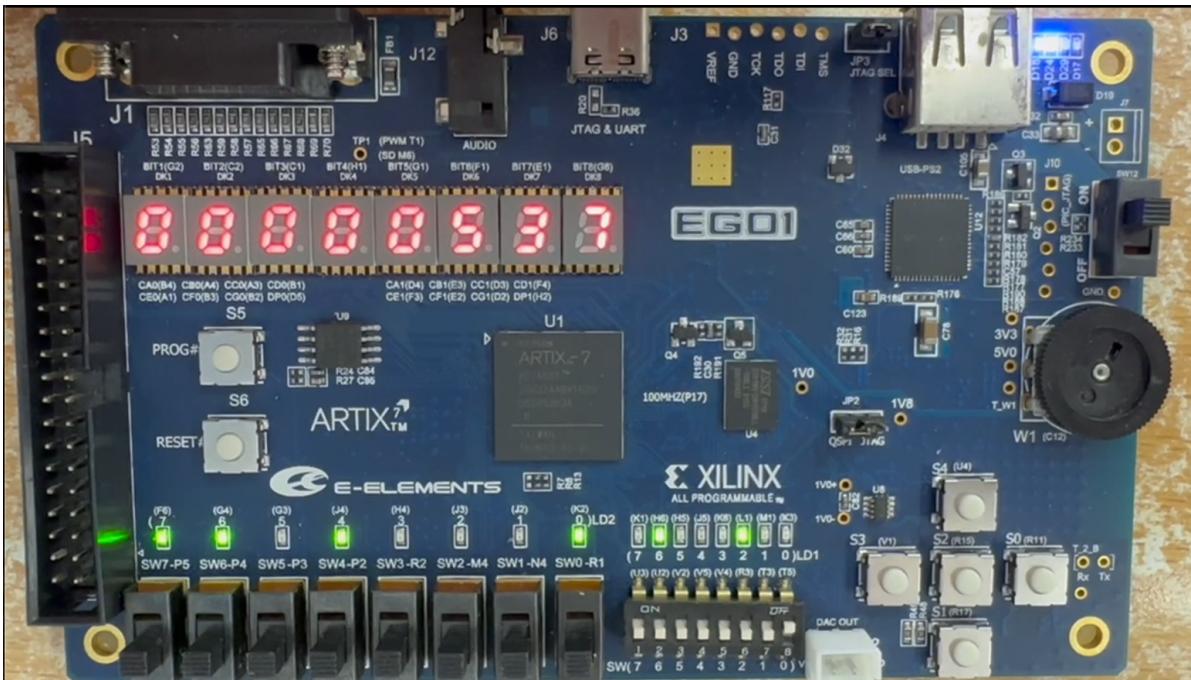


图5: 板级验证示例-译码lui x10, 0

目前的演示中将zero值设置为1。下方16个led从左到右的含义为：

```
inst[0], z, wmem, wreg, m2reg, aluc[3:0] , aluimm, pcsource[1:0], sext, i_lui,  
i_sw, shift
```

上图中译码lui x10, 0。lui x10, 0的机器码的十六进制表示为00000537，对应的值如下：

表1: lui x10, 0产生的控制信号表

z	wmem	wreg	m2reg	aluc[3:0]	aluimm	pcsource[1:0]	sextr	i_lui	i_sw
1	0	1	0	0010	1	00	0	1	0

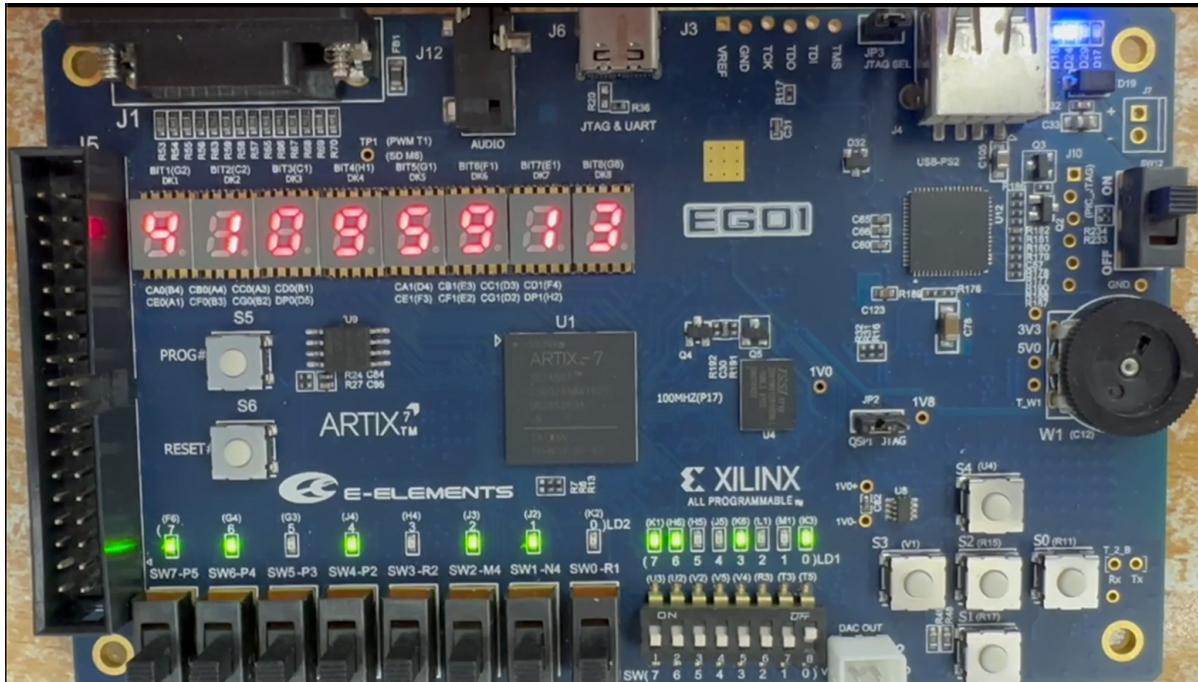


图5: 板级验证示例-译码srai x18, x18, 16

表2: srai x18, x18, 16产生的控制信号表

z	wmem	wreg	m2reg	aluc[3:0]	aluimm	pcsource[1:0]	sext	i_lui	i_sw
1	0	1	0	1101	1	00	1	0	0

Appendix

板级验证视频附在压缩包内。