Computation II: embedded system design (5EIBO)

Dashbo... / My cou... / 5El... / Practice Assessment 2024-... / Part 2a: 2024-04-05 Mini MIPS Adding Custom Instructions (Binarization Calc...

Available from: Friday, 5 April 2024, 2:00 PM

Requested files: ctrl.v, aluctrl.v, alu.v (Download)

Type of work:
Individual work

Introduction

Image binarization converts an image of up to 256 gray levels to a black and white image. The simplest way to use image binarization is to choose a threshold value, and classify all pixels with values above this threshold as white (255), and all other pixels as black (0). The C code below handles this with an if-statement resulting in many processor instructions.

```
#define WIDTH     32
#define HEIGHT     32

void main(void)
{
    int a, b, result;
    int threshold = 128;
    unsigned char *buf_i = (unsigned char*)0x401000, *buf_o = (unsigned char*)0x402000;

    for (a = 1; a < HEIGHT - 1; a++)
    {
        for (b = 1; b < WIDTH - 1; b++)
        {
            result = buf_o[a * WIDTH + b];
            /* Thresholding */
            if(result > 128) buf_o[a * WIDTH + b] = (char)255;
            else buf_o[a * WIDTH + b] = 0;

        }
    }
}
```

To reduce the number of instructions, the C code is modified to use a custom instruction (thresholding), giving the following C code:

```
#define WIDTH     32
#define HEIGHT     32
#define thresholding(result, threshold) ((result) - ((threshold) + *(int *) 0x12344321)) // HEX Code: 0x31

void main(void)
{
    int a, b, result;
    int threshold = 128;
    unsigned char *buf_i = (unsigned char*)0x401000, *buf_o = (unsigned char*)0x402000;

    for (a = 1; a < HEIGHT - 1; a++)
    {
        for (b = 1; b < WIDTH - 1; b++)
        {
            result = buf_o[a * WIDTH + b];
            /* Thresholding */
            buf_o[a * WIDTH + b] = thresholding(result, threshold);
        }
    }
}</pre>
```

Add hardware support for the custom instruction (**thresholding**) in the current mMIPS implementation by modifying all or some of the provided files (ctrl.v, aluctrl.v and alu.v). You can infer the functionality of **thresholding** by comparing the C codes above.

Note: The thresholding function code is 0x31.

Debug

For debugging the design, you can use the "\$display" command in the RTL code to print out the values of interest. Note: "\$display" will only work in the debug mode of VPL.

Requested files

ctrl.v

```
// CTRL.V
  4
     // TU/e Eindhoven University Of Technology
  5
     // Eindhoven, The Netherlands
  6
     // Created: 21-11-2013
     // Author: Bergmans, G (g.bergmans@student.tue.nl)
// Based on work by Sander Stuijk
  8
 10
 11
     // Function:
     //
 12
             Single cycle controller
 13
 14
 15
     //
             (27-01-2014): initial version
     //
 16
 17
     19
20
     module CTRL(
              enable,
 21
              en,
 22
              Opcode.
 23
              FunctionCode.
 24
              RegDst,
 25
              Target,
 26
              Branch
 27
              MemRead,
 28
              ALUop,
MemWrite,
 29
 30
 31
              ALUSrc,
 32
              RegWrite,
33
34
              SignExtend,
              c4,
 35
              c1,
36
37
              c31,
HiLoWrite,
 38
              AluSel
 39
         );
 40
 41
                  enable;
         input
 42
         output
                  [0:0]
43
44
          reg
                  [0:0]
                           en;
          input
                          Opcode:
                  [5:0]
 45
                  [5:0]
                          FunctionCode;
          input
 46
          output
                  [1:0]
                           RegDst;
 47
          reg
                  [1:0]
                          RegDst;
 48
         output
                  [1:0]
                           Target;
 49
                  [1:0]
          reg
                           Target;
          output
 50
                  [1:0]
                          Branch;
 51
                          Branch:
          reg
                  [1:0]
 52
         output
                          MemRead;
                  [1:0]
 53
54
          reg
                  [1:0]
                          MemRead;
         output
                  [1:0]
                          MemtoReg:
 55
                          MemtoReg;
                  [1:0]
          reg
56
57
          output
                  [4:0]
                           ALUop;
          reg
                  [4:0]
                          ALUop;
         output
 58
                          MemWrite;
                  [1:0]
 59
          reg
                  [1:0]
                          MemWrite;
 60
          output
                  [0:0]
                          ALUSrc;
 61
         reg
output
                  [0:0]
                          ALUSrc:
                  [0:0]
 62
                           RegWrite;
 63
          reg
                  [0:0]
                           RegWrite;
          output
 64
                  [0:0]
                          SignExtend;
 65
                  [0:0]
                          SignExtend;
          reg
 66
          output
                  [31:0]
                          c4;
 67
          reg
                  [31:0]
                          c4;
 68
         output
                  [0:0]
                          c1;
 69
                  [0:0]
                          c1;
          reg
 70
          output
                  [4:0]
                          c31;
 71
72
                  [4:0]
[0:0]
          reg
                           c31;
         output
                          HiLoWrite;
 73
                  [0:0]
                          HiLoWrite;
          reg
 74
75
          output
                  [1:0]
                          AluSel;
         reg
                  [1:0]
                          AluSel:
 76
 77
78
79
         always @(Opcode or FunctionCode or enable)
              begin
 80
                  81
 82
                  //Write constant 1 to output
 84
                  c1 = 1'b1;
                  //Write constant 31 to output c31 = 5'b11111;
 85
 86
 87
 88
                  if (enable == 1)
                      en = 1'b1;
 89
 90
 91
                      en = 1'b0;
 92
 93
                  RegDst
                               = 2'b00;
 94
                               = 2'b00;
                  Target
                               = 1'b0;
= 2'b00;
 95
                  ALUSrc
 96
                  MemtoReg
 97
                  RegWrite
                                 1'b0;
98
99
                               = 2'b00;
= 2'b00;
                  MemRead
                  MemWrite
                               = 2'b00;
100
                  Branch
101
                  ALUop
                                 5'b00000;
102
                  SignExtend
                              = 1'b0;
```

```
104
                   //Determine the output
105
                    case (Opcode)
                   0: // R-format instruction: check functioncode
106
                        case (FunctionCode)
'h8: // Instruc
107
108
                                   // Instruction: Jr
109
                                 begin
110
                                     RegDst
                                                     2'b01:
                                                     2'b10;
                                      Target
111
112
                                      ALUSrc
                                                     1'b0;
                                      MemtoReg
                                                     2'b00;
113
                                                     1'b0;
114
                                      RegWrite
115
                                      MemRead
                                                     2'b00;
116
                                      MemWrite
                                                     2'b00;
                                                     2'b11;
117
                                     Branch
                                                     5'b00010;
118
                                     ALUop
119
                                      SignExtend
                                                     1'b1;
120
                                     HiLoWrite
                                                   =
                                                     1'h0:
                                                   = 2'b00;
121
                                     AluSel
122
                                 end
123
                             'h9:
                                   // Instruction Jalr
                                begin
124
125
                                     RegDst
                                                     2'b01;
                                                     2'b10;
1'b0;
126
                                      Target
127
                                     ALUSTO
128
                                                     2'b00;
                                      MemtoReg
129
                                     RegWrite
                                                     1'b1;
130
                                      MemRead
                                                     2'b00
                                                     2'b00;
131
                                      MemWrite
                                                     <mark>2</mark>'b11;
132
                                     Branch
                                                   = 5'b00010;
= 1'b1;
133
                                      ALUop
                                     SignExtend
134
135
                                     HiLoWrite
                                                   = 1'b0;
136
                                                   = 2'b11;
                                      AluSel
137
                                 end
                             'h10: // Instruction: Move hi register
138
                                begin
139
140
                                     RegDst
                                                   = 2'b01;
                                                   = 2'b00;
141
                                      Target
142
                                                     1'b0;
                                      ALUSrc
143
                                      MemtoReg
                                                     2'b00;
144
                                     RegWrite
                                                     1'b1;
                                                     2'b00;
145
                                      MemRead
                                                     2'b00;
146
                                      MemWrite
147
                                     Branch
                                                     2'b00
                                                     5'b00010;
148
                                     ALUop
149
                                     SignExtend
                                                   = 1'b1;
                                      HiLoWrite
150
151
                                     AluSel
                                                   = 2'b10;
152
                                 end
153
                             'h12:
                                    // Instruction: Move lo register
154
                                 begin
155
                                     RegDst
                                                   = 2'b01;
                                                     2'b00;
156
                                      Target
157
                                      ALUSrc
                                                     1'b0;
158
                                      MemtoReg
                                                     2'b00;
                                                     1'b1:
159
                                     RegWrite
160
                                                     2'b00
                                      MemRead
161
                                     MemWrite
                                                     2'b00
                                                     2'b00:
162
                                     Branch
                                                     5'b00010;
                                     ALUop
163
164
                                      SignExtend
                                                     1'b1;
165
                                     HiLoWrite
                                                   = 1'b0;
= 2'b01;
                                     AluSel
166
167
                                 end
168
                             'h19: // Instruction: Multiply unsigned
169
                                 begin
170
                                     RegDst
                                                   = 2'b00; //No destination
171
                                      Target
                                                   = 2'b00;
                                                   = 1'b0;
= 2'b00;
172
                                      ALUSrc
173
                                      MemtoReg
                                      RegWrite
174
                                                     1'b1;
                                                     2'b00;
2'b00;
175
                                      MemRead
176
                                      MemWrite
177
                                                     2'b00;
                                     Branch
178
                                     ALUop
                                                     5'b00010;
179
                                     SignExtend
                                                   =
                                                     1'b1;
                                     HiLoWrite
                                                     1'b1;
180
181
                                     AluSel
                                                     2'b00;
182
                                 end
                            default: // Others
183
184
                                 begin
185
                                     RegDst
                                                   = 2'b01;
= 2'b00;
186
                                      Target
                                                     1'b0;
187
                                      ALUSrc
188
                                      MemtoReg
                                                     2'b00;
                                                     1'b1;
2'b00;
189
                                     RegWrite
                                      MemRead
190
191
                                      MemWrite
                                                     2'b00;
192
                                     Branch
                                                     2'b00
                                                     5'b00010;
193
                                     ALUop
194
                                     SignExtend
                                                     1'b1;
                                                   = 1'b0;
= 2'b00;
195
                                     HiLoWrite
196
                                     AluSel
197
                                      end
198
                            endcase
199
                        // Instruction: J
200
                        begin
201
                            RegDst
                                          = 2'b00;
202
                                          = 2'b01;
                             Target
203
                            ALUSrc
                                          = 1'b0;
204
                                          = 2'b00;
                            MemtoReg
                                          = 1'b0;
205
                            RegWrite
206
                            MemRead
                                          = 2'b00
```

```
= 2'b00;
= 2'b11;
207
                             memwrite
208
                             Branch
209
                             ALUop
                                             5'b00010;
                                           = 1'b1;
= 1'b0;
210
                             SignExtend
211
                             HiLoWrite
                                           = 2'b00;
212
                             AluSel
213
214
                    3:
                        // Instruction; Jal
215
                        begin
216
                             RegDst
                                             2'b10;
                                           = 2'b01;
= 1'b0;
217
                             Target
218
                             ALUSrc
219
                             MemtoReg
                                             2'b00
220
                             RegWrite
                                             1'b1;
221
                             MemRead
                                           = 2'b00;
                                             2'b00;
                             MemWrite
222
223
                                             2'b11;
                             Branch
                                           = 5'b00010;
= 1'b1;
224
                             ALUop
225
                             SignExtend
226
                             HiLoWrite
227
                             AluSel
                                           = 2'b11;
228
                        end
                        // Instruction: BEQ
229
230
                        begin
231
                             RegDst
                                           = 2'b00;
                                           = 2'b00;
232
                             Target
ALUSrc
                                             1'b0;
233
                             MemtoReg
234
                                               'b00;
235
                             RegWrite
                                             1'b0;
                                             2'b00;
236
                             MemRead
237
                             MemWrite
                                             2'b00;
238
                             Branch
                                             2'b01
                                             5'b00001;
239
                             ALUop
240
                             SignExtend
                                             1'b1;
241
                             HiLoWrite
                                             1'b0;
                                           = 2'b00;
242
                             AluSel
243
                        end
244
                        // Instruction: BNE
245
                        begin
                                           = 2'b00;
= 2'b00;
246
                             RegDst
247
                             Target
248
                             ALUSrc
                                               'b0;
249
                                             2'h00:
                             MemtoReg
250
                             RegWrite
                                             1'b0;
                                            2'b00;
2'b00;
251
                             MemRead
252
                             MemWrite
                                             2'b10;
253
                             Branch
                                               'b00001;
254
                             ALUop
                                           = 1'b1;
= 1'b0;
255
                             SignExtend
256
                             HiLoWrite
257
                             AluSel
                                           = 2'b00;
258
                        // Instruction: ADDIU
259
260
                        begin
                             RegDst
261
                                           = 2'b00;
= 1'b1;
262
                             Target
263
                             ALUSrc
264
                             MemtoReg
                                             2'b00;
265
                             RegWrite
                                             1'b1;
266
                             MemRead
                                             2'b00
                                             2'b00;
267
                             MemWrite
268
                             Branch
                                               b00;
269
                             ALUop
                                           = 5'b00011;
= 1'b1;
270
                             SignExtend
271
                             HiLoWrite
                                             1'b0;
272
                             AluSel
                                           = 2'b00
273
                        end
274
                    10:
                         // Instruction: SLTI
275
                        begin
276
                             RegDst
                                           = 2'b00;
                                           = 2'b00;
277
                             Target
ALUSrc
278
                                             1'b1;
                             MemtoReg
279
                                           = 2'b00
                                           = 1'b1;
= 2'b00;
280
                             RegWrite
281
                             MemRead
282
                             MemWrite
                                             2'b00;
283
                             Branch
                                             2'b00:
                                             5'b00111;
284
                             ALUop
285
                             SignExtend
                                             1'b1;
                             HiLoWrite
286
                                             1'b0;
                                           = 2'b00
287
                             AluSel
288
                        end
289
                         // Instruction: SLTUI
290
                        begin
                             RegDst
291
                                           = 2'b00:
292
                                           = 2'b00;
                             Target
293
                             ALUSrc
                                             1'b1;
294
                             MemtoReg
                                           = 2'b00;
                                             1'b1;
295
                             RegWrite
296
                             MemRead
                                             2'b00;
297
                             MemWrite
                                             2'b00
                                             2'b00;
298
                             Branch
299
                             ALUop
                                             5
                                               'b01000;
300
                             SignExtend
                                             1'b1;
                                             1'h0:
301
                             HiLoWrite
                                           = 2'b00;
302
                             AluSel
303
304
                    12:
                         // Instruction: ANDI
305
                        begin
306
                             RegDst
                                           = 2'b00;
                                           = 2'b00;
= 1'b1;
307
                             Target
308
                             ALUSTO
                                           = 2'b00;
= 1'b1:
309
                             MemtoReg
```

```
= 2'b00;
= 2'b00;
311
                             MemRead
312
                             MemWrite
313
                                             2'b00;
                             Branch
314
                             ALUop
                                            5'b00100;
                                          = 1'b0;
315
                             SignExtend
                                          = 1'b0;
                             HiLoWrite
316
317
                                          = 2'b00;
318
                        end
                    13:
                         // Instructino: ORI
319
320
                        begin
321
                             RegDst
                                          = 2'b00;
= 2'b00;
                            Target
ALUSrc
322
                                          = 1'b1;
323
324
                                             2'b00;
                             MemtoReg
                                            1'b1;
2'b00;
325
                             RegWrite
326
                             MemRead
327
                             MemWrite
                                             2'b00;
                                            2'b00;
5'b00101;
328
                             Branch
329
                             ALUop
330
                                            1'b0;
                             SignExtend
331
                             HiLoWrite
332
                             AluSel
                                          = 2'b00;
333
                        end
                         // Instruction: XORI
334
335
                        begin
                             RegDst
                                          = 2'b00:
336
                                          = 2'b00;
337
                             Target
                                             1'b1;
338
339
                             MemtoReg
                                            2'b00
                                            1'b1;
340
                             RegWrite
341
                                             2'b00;
                             MemRead
                                          = 2'b00;
= 2'b00;
342
                             MemWrite
343
                             Branch
344
                                            5'b00110;
                             ALUop
345
                             SignExtend
                                            1'b0;
346
                             HiLoWrite
                                          = 1'b0;
347
                             AluSel
                                          = 2'b00:
348
                        end
349
                    15:
                         // Instruction: LUI
350
                        begin
351
                             RegDst
                                             2'b00;
352
                             Target
                                             2'b00;
353
                             ALUSrc
                                          = 1'b1;
                             MemtoReg
                                             2'b00;
354
355
                             RegWrite
                                             1'b1;
356
                             MemRead
                                          = 2'b00
                                            2'b00;
357
                             MemWrite
                             Branch
                                               'b00;
358
359
                             ALUop
                                             5'b01001;
                                          = 1'b1;
360
                             SignExtend
                                            1'b0;
361
                             HiLoWrite
362
                                          = 2'b00
                             AluSel
363
                        end
                    32:
                         //Instruction: LB
364
365
                        begin
366
                             RegDst
                                            2'b00;
367
                             Target
                                          = 2'b00;
                                          = 1'b1;
368
                             ALUSrc
369
                             MemtoReg
                                             2'b10;
370
                             RegWrite
                                            1'b1;
2'b10;
371
                             MemRead
                             MemWrite
                                            2'b00;
372
373
                             Branch
                                             2'b00;
                                            5'b00000:
374
                             ALUop
                                          = 1'b1;
375
                             SignExtend
376
                             HiLoWrite
                                          = 1'b0;
377
                             AluSel
                                          = 2'b00;
378
                        end
379
                    35:
                         // Instruction: LW
380
                        begin
                             RegDst
381
                                          = 2'h00:
                                          = 2'b00;
382
                             Target
383
                             ALUSrc
                                            1'b1;
384
                             MemtoReg
                                            2'b01;
1'b1;
385
                             RegWrite
                                             2'b01;
386
                             MemRead
387
                             MemWrite
                                          = 2'b00;
= 2'b00;
388
                             Branch
                                            5'b00000;
389
                             ALUop
390
                                          = 1'b1;
                             SignExtend
                                          = 1'b0;
= 2'b00;
391
                             HiLoWrite
392
                             AluSel
393
                        end
394
                    40:
                         // Instruction: SB
395
                        begin
396
                             RegDst
                                          = 2'b00;
397
                                          = 2'b00;
                             Target
                                          = 1'b1;
398
                             ALUSTO
                                          = 2'b00;
399
                             MemtoReg
400
                             RegWrite
                                            1'b0;
                                          = 2'b00;
= 2'b10;
401
                             MemRead
                             MemWrite
402
403
                             Branch
                                               'b00;
404
                             ALUop
                                             5'b00000;
405
                             SignExtend
                                          = 1'b1;
                                          = 1'b0;
406
                             HiLoWrite
407
                                          = 2'b00;
                             AluSel
408
                        end
                    43:
409
                         // Instruction: SW
410
                        begin
411
                             RegDst
                                             2'b00;
                                          = 2'b00;
= 1'h1:
412
                             Target
```

ALUSTO

413

```
= 2'b00;
= 1'b0;
= 2'b00;
= 2'b01;
= 2'b00;
= 5'b00000;
= 1'b1;
= 1'b0;
= 2'b00;
                                              MemtoReg
                                              RegWrite
MemRead
MemWrite
415
416
417
418
                                               Branch
419
420
                                              ALUop
SignExtend
HiLoWrite
421
422
423
424
                                               AluSel
                                end
default: //No default case
425
                                       begin
426
427
                                       end
                                endcase
428
                         end
429
430
          endmodule
431
```

aluctrl.v

```
// ALUCTRL.V
  4
     // TU/e Eindhoven University Of Technology
     // Eindhoven, The Netherlands
  6
     // Created: 21-11-2013
     // Author: Bergmans, G (g.bergmans@student.tue.nl)
// Based on work by Sander Stuijk
  8
 10
 11
     // Function:
     //
 12
            ALU controller
 13
 15
     //
            (27-01-2014): initial version
     //
 16
 17
     19
     module ALUCTRL(functionCode, ALUop, Shamt, ALUctrl);
                          functionCode;
 20
         input
                 [5:0]
[4:0]
 21
          input
                 [4:0]
[5:0]
 22
         input
                          Shamt;
 23
                          ALUctrl
         output
                  [5:0]
                          ALUctrl;
         reg
 25
 26
         always @(functionCode or ALUop or Shamt)
              begin : aluctrl_thread
 27
                  case (ALUop) //synopsys parallel_case
                          : // Add signed
ALUctrl = 'h2;
 29
                       'h0:
 30
 31
                      'h1:
                               // Subtract unsigned
                          ALUctrl = 'h6;
 33
34
 35
                      'h2:
                               // R-type instruction, look to functionCode
36
37
                               case (functionCode)
 38
                                    'h0:
                                           // SLL
 39
                                       case (Shamt) //Check shift amount
 40
                                           1:
 41
                                               ALUctrl = 'hA;
                                           2:
43
44
                                               ALUctrl = 'hB;
                                           8:
 45
                                                ALUctrl = 'hC;
 46
                                           default
                                               ALUctrl = 'h0;
 47
 48
                                       endcase
 49
 50
                                   'h2:
                                           // SRI
                                       case (Shamt) //Check shift amount
 51
 52
                                           1
53
54
                                                ALUctrl = 'hD;
                                           2:
 55
                                               ALUctrl = 'hE;
56
57
                                               ALUctrl = 'hF;
 58
                                           default:
 59
                                               ALUctrl = 'h0;
 60
                                       endcase
 61
 62
                                   'h3:
                                           // SRA
                                       case (Shamt) //Check shift amount
 64
                                           1:
 65
                                               ALUctrl = 'h10;
                                           2:
 67
                                                ALUctrl = 'h11;
 68
                                           8:
                                                ALUctrl = 'h12;
 70
                                           default:
71
72
73
74
                                               ALUctrl = 'h0;
                                       endcase
                                   'h10: // Move hi register (nop in ALU)
ALUctrl = 'h0;
 75
 76
 77
                                   'h12: // Move hi register (nop in ALU)
78
79
                                       ALUctrl = 'h0;
 80
                                   'h19: // Multiply unsigned
                                       ALUctrl = 'h13;
 81
 82
                                   'h20: // Add signed
 84
                                       ALUctrl = 'h2;
 85
                                   'h21: // Add unsigned
ALUctrl = 'h3;
 86
 87
 88
                                           // Subtract unsigned
 89
                                   'h23:
                                       ALUctrl = 'h6;
 91
 92
                                       4: // And
ALUctrl = 'h0;
                                   'h24:
 93
                                       5: // Or
ALUctrl = 'h1;
 95
                                   'h25:
 96
 97
                                   'h26: // Xor
ALUctrl = 'h4;
98
99
100
101
                                           //Set-on-less-than (2's complement)
102
                                       ALUctrl = h7;
103
```

```
'h2B: //Set-on-less-than (unsigned)
ALUctrl = 'h8;
104
105
106
107
                                       default:
108
                                            ALUctrl = 'h0;
109
                                   endcase
                              end: // Add unsigned
ALUctrl = 6'b000011;
110
                          'h3:
111
112
113
114
                                 // And
                          'h4:
                              ALUctrl = 6'b000000;
115
116
                                // Or
                         'h5:
117
118
                              ALUctrl = 6'b000001;
119
                              : // Xor
ALUctrl = 6'b000100;
                         'h6:
120
121
122
123
124
                              : //Slt
ALUctrl = 6'b000111;
                          'h7:
125
126
                          'h8:
                                  //Sltu
                              ALUctrl = 6'b001000;
127
128
                              : //Load upper immediate
ALUctrl = 6'b001001;
129
130
131
132
                    ALUctrl = 6'b000000; endcase
                         default:
133
134
135
136
      endmodule
137
138
```

alu.v

```
// ALU.V
  4
     // TU/e Eindhoven University Of Technology
  5
     // Eindhoven, The Netherlands
  6
     // Created: 21-11-2013
     // Author: Bergmans, G (g.bergmans@student.tue.nl)
// Based on work by Sander Stuijk
  8
 10
 11
     // Function:
     //
 12
           Arithmetic Logic Unit
 13
 15
     //
            (27-01-2014): initial version
     //
 16
 17
     module ALU(ctrl, a, b, r, r2, z);
  input    [5:0]    ctrl;
  input    [31:0]    a;
 19
 20
 21
 22
         input
                      [31:0]
 23
                      [31:0]
         output
                              r;
                      [31:0]
          reg
 25
          output
                      [31:0]
 26
          reg
                       [31:0] r2;
         output
 27
                      [0:0]
                              z;
 28
                      [0:0]
          reg
                               z;
 29
          reg
                       [31:0]
 30
          reg
                      [31:0]
                              t;
                      [31:0]
                              s_int;
t_int;
 31
          reg signed
 32
          reg signed
                      [31:0]
 33
34
          reg
                       [31:0]
                              result;
                      [31:0]
                              result hi;
         reg
 35
          reg
                      [0:0]
                               sign;
36
37
          reg signed
                      [63:0] c;
         reg
                      [0:0]
                              zero:
 38
 39
         always @(ctrl or a or b)
              begin : alu_thread
 40
 41
                  //Read the inputs
                              = a;
= b;
 43
                  s
t
 44
 45
                  s_int
                               = s;
 46
                  t_int
                              = t;
 47
                  result
                               = 0:
 48
                  result hi = 0;
 49
                  // Calculate result using selected operation
 50
 51
                  case (ctrl)
  'h0: // And
 52
 53
54
                          result = s & t;
                      'h1: // Or
 55
                          result = s | t;
 57
                      'h2: // Add signed
 58
 59
                          result = s_int + t_int;
 60
                      'h3: // Add unsigned
 61
 62
                          result = s + t;
 63
                      'h4: // Xor
 64
                          result = s ^ t;
 65
 66
                      'h6: // Substract signed
 67
 68
                          result = s - t;
 69
 70
                              // Set-on-less-than
                          if (s_int < t_int)
result = 1;
71
72
 73
74
                          else
                               result = 0;
 75
 76
                      'h8:
                               // Set-on-less-than unsigned
 77
                          if (s < t)
78
79
                              result = 1;
                          else
 80
                               result = 0;
 81
                               // Load upper immediate
                      'h9:
 82
                          result = (t << 16);
 84
                          : // SLL (1 bit) result = (t << 1);
                      'hA:
 85
 86
 87
 88
                      'hB:
                              // SLL (2 bit)
 89
                          result = (t << 2);
 90
 91
                      'hC:
                              // SLL (8 bit)
 92
                          result = (t << 8);
 93
                               // SRL (1 bit)
                      'hD:
 95
                          result = (t >> 1);
 96
 97
                      'hE: // SRL (2 bit)
98
99
                          result = (t >> 2);
100
                       'hF:
                              // SRL (8 bit)
101
                          result = (t >> 8);
102
                      'h10:
                              // SRA (1 bit)
103
```

```
104
                                begin
                                     sign = t[31:31];
105
                                     result = (t >> 1);
result[31:31] = sign;
106
107
108
109
                               1: // SRA (2 bit)
begin
                           'h11:
110
111
                                    sign = t[31:31];
result = (t >> 2);
result[31:30] = {sign, sign};
112
113
114
115
116
                           'h12:
                                    // SRA (8 bit)
117
118
                               begin
                                    sign = t[31:31];
result = (t >> 8);
result[31:24] = {sign, sign, sign, sign, sign, sign, sign};
119
120
121
122
123
                           'h13:
                                    //Multu
124
125
                               begin
                                    c = s * t;
result = c[31:0];
result_hi = c[63:32];
126
127
128
129
130
                           default: //No default case: invallid opcode!
131
132
                               begin
133
                      endcase
134
135
136
                      // Calculate zero output
                      if (result == 0)
zero = 1;
137
138
139
                      else
140
                           zero = 0;
141
142
                      // Write results to output
                      r = result;
r2 = result_hi;
143
144
145
                      z = zero;
146
147
148
      endmodule
149
```

VPL

You are logged in as Thomas Stirling Valdez (Log out) 5EIBO

Data retention summary