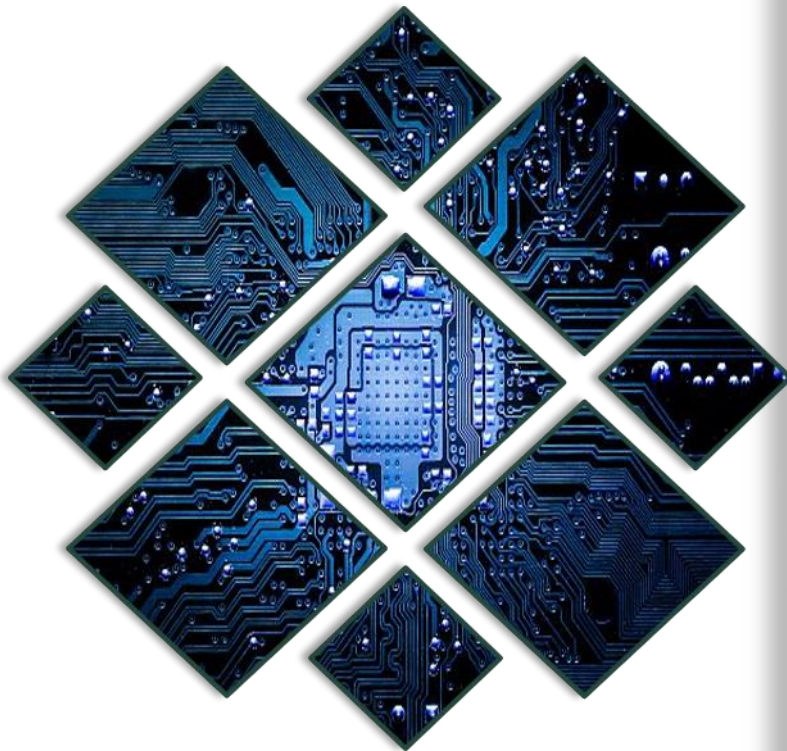


VLSI

Laboratory 4: Controller Design



Correal Ramos, Victor

Rodríguez Martínez, Carlos

Q1 2020-21

LAB4: CONTROLLER DESIGN

1. Hours

We spend 8h more or less for this laboratory.

2. Verifications's Tools

2.1 std_nand3

DRC

```
===== 2 =====  
Found 9 networks, allocated 32 bytes for network numbering (took 0.003 seconds)  
0 errors found (took 0.007 seconds)  
===== 3 =====
```

ERC

```
===== 3 =====  
Checking Wells and Substrates... (type Windows-C to abort)  
3 nodes of type P-Transistor  
3 nodes of type N-Transistor  
9 networks found  
No Well errors found (0.028 seconds)
```

NCC

```
===== 3 =====  
Checking Wells and Substrates... (type Windows-C to abort)  
3 nodes of type P-Transistor  
3 nodes of type N-Transistor  
9 networks found  
No Well errors found (0.028 seconds)
```

2.2 Alucontrol

DRC

```
===== 3 =====  
Found 22 networks, allocated 232 bytes for network numbering (took 0.002 seconds)  
Checking facet alucontrol{lay}  
No errors found (0.047 seconds so far)  
0 errors found (took 0.05 seconds)
```

ERC

```
===== 6 =====  
Checking Wells and Substrates... (type Windows-C to abort)  
13 nodes of type P-Transistor  
13 nodes of type N-Transistor  
22 networks found  
No Well errors found (0.037 seconds)
```

```

===== 7 =====
Comparing facet std_inv{lay} with facet std_inv{sch}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from std_inv{lay}...
Extracting networks from std_inv{sch}...
Both facets have 2 components
Both facets have 2 nets
Facets std_inv{lay} and std_inv{sch} are equivalent (0.018 seconds)
Comparing facet std_nor2{lay} with facet std_nor2{sch}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from std_nor2{lay}...
Extracting networks from std_nor2{sch}...
Both facets have 4 components
Both facets have 4 nets
Facets std_nor2{lay} and std_nor2{sch} are equivalent (0.032 seconds)
Comparing facet std_nand2{lay} with facet std_nand2{sch}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from std_nand2{lay}...
Extracting networks from std_nand2{sch}...
Both facets have 4 components
Both facets have 4 nets
Facets std_nand2{lay} and std_nand2{sch} are equivalent (0.046 seconds)
Comparing facet alucontrol{lay} with facet alucontrol{sch}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from alucontrol{lay}...
Extracting networks from alucontrol{sch}...
Both facets have 8 components
Both facets have 14 nets
Facets alucontrol{lay} and alucontrol{sch} are equivalent (0.06 seconds)

```

2.3 Controller

DRC

```

===== 8 =====
Found 233 networks, allocated 2820 bytes for network numbering (took 0.003 seconds)
0 errors found (took 0.007 seconds)
===== 9 =====

```

ERC

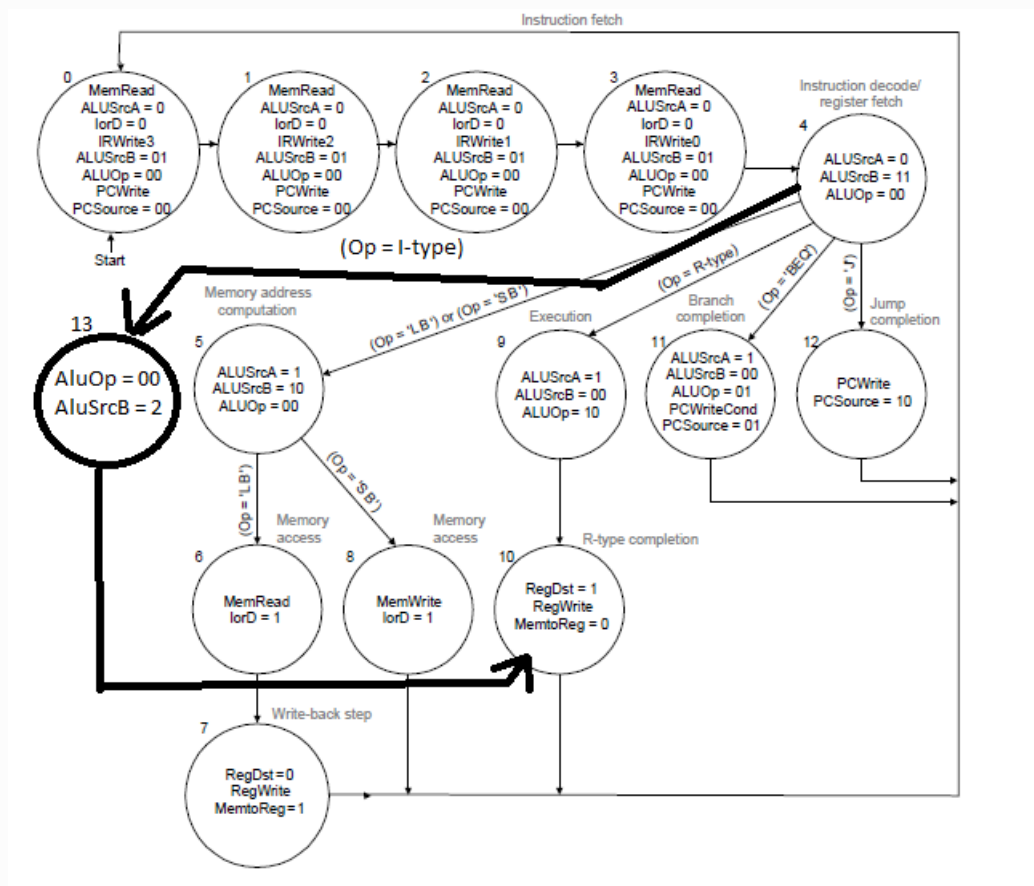
```

===== 9 =====
Checking Wells and Substrates... (type Windows-C to abort)
220 nodes of type P-Transistor
220 nodes of type N-Transistor
233 networks found
No Well errors found (0.04 seconds)

```

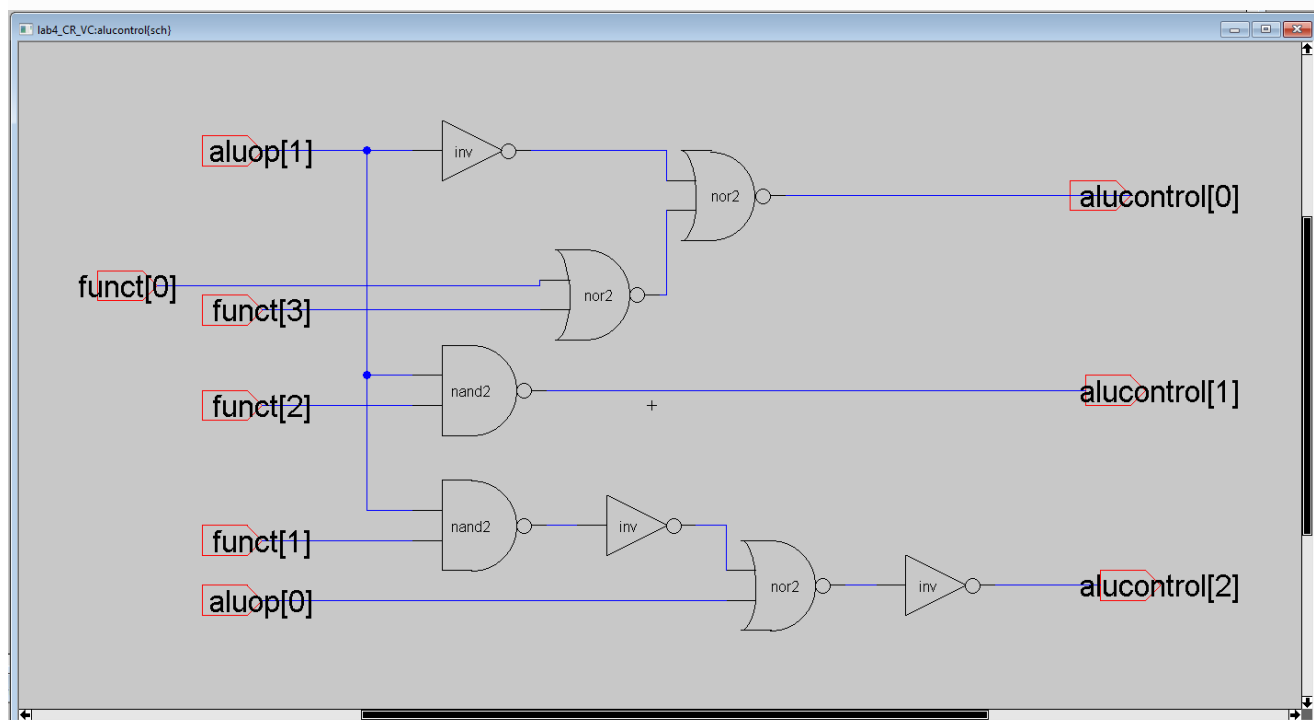
```
===== 10 =====
Comparing facet std_nor3{sch} with facet std_nor3{lay}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from std_nor3{sch}...
Extracting networks from std_nor3{lay}...
Both facets have 6 components
Both facets have 6 nets
Facets std_nor3{sch} and std_nor3{lay} are equivalent (0.012 seconds)
Comparing facet std_latch{sch} with facet std_latch{lay}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from std_latch{sch}...
Extracting networks from std_latch{lay}...
Both facets have 12 components
Both facets have 8 nets
Facets std_latch{sch} and std_latch{lay} are equivalent (0.022 seconds)
Comparing facet controller{sch} with facet controller{lay}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No facet overrides
Extracting networks from controller{sch}...
Extracting networks from controller{lay}...
Both facets have 95 components
Both facets have 105 nets
Facets controller{sch} and controller{lay} are equivalent (0.033 seconds)
```

3. FSM diagram

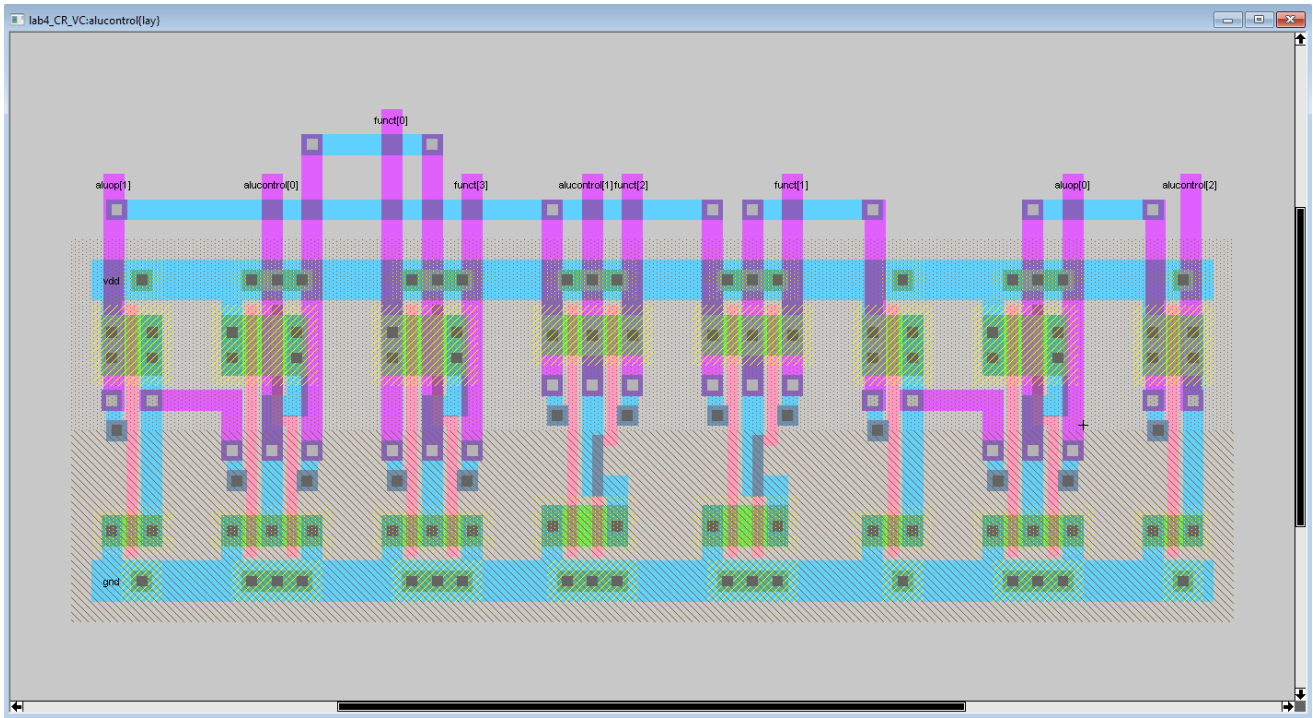


4. Printout alucontrol

4.1 Schematic



4.2 Layout



5. Printout alucontrol.cmd

We do without assertion failures. We test ADD, SUB, AND, OR and SLT without problems. It's works well.

```
1 | alucontrol.cmd
2 | Written 2/12/02 David_Harris@hmc.edu
3 |
4 | Simulate the alucontrol block.
5 | only consider legal instructions
6 | Note: this file is incomplete
7 |
8 vector aluop aluop[1] aluop[0]
9 vector func func[3] func[2] func[1] func[0]
10 vector alucontrol alucontrol[2] alucontrol[1] alucontrol[0]
11
12 | test always adding
13 set aluop 00
14 s 2
15 assert alucontrol 010
16
17 | test always subtracting
18 set aluop 01
19 s 2
20 assert alucontrol 110
21
22 | test func ADD
23 set aluop 10
24 set func 0000
25 s 2
26 assert alucontrol 010
27
28 | add code here to test SUB, AND, OR, SLT
29
30 | test always SUB
31 set aluop 10
```

```

32  set funct 0010
33  s 2
34  assert alucontrol 110
35
36  | test always AND
37  set aluop 10
38  set funct 0100
39  s 2
40  assert alucontrol 000
41
42  | test always OR
43  set aluop 10
44  set funct 0101
45  s 2
46  assert alucontrol 001
47
48  | test always SLT
49  set aluop 10
50  set funct 1010
51  s 2
52  assert alucontrol 111

```

6. Controller.cmd

The simulation with the file controller.cmd was good. Any problems appear.

```

1  | controller.cmd
2  | Written 2/12/02 David_Harris@hmc.edu
3  |
4  | Simulate the controller for lab 4.
5
6  | define vectors for convenience
7  vector op op5 op4 op3 op2 op1 op0
8  vector irwrite irwrite3 irwrite2 irwrite1 irwrite0
9  vector aluop aluop1 aluop0
10 vector alusrcb alusrcb1 alusrcb0
11 vector pcsource pcsource1 pcsource0
12
13 | define a 2-phase nonoverlapping clock
14 | in which ph1 goes high then low
15 | then ph2 goes high then low
16 | with a time of 2 ns for each step (8 ns cycle time)
17 clock ph1 1 0 0 0
18 clock ph2 0 0 1 0
19 stepsize 2
20
21 | reset the system by setting reset high and clocking twice
22 h reset
23 l zero
24 c
25 l reset
26
27 | ***** test lb
28 | state 0
29 set op 010000

```

```
30 c
31 assert memread 1
32 assert alusrca 0
33 assert iord 0
34 assert irwrite 1000
35 assert alusrcb 01
36 assert aluop 00
37 assert pcchange 1
38 assert pcsource 00
39
40 | state 1
41 c
42 assert memread 1
43 assert alusrca 0
44 assert iord 0
45 assert irwrite 0100
46 assert alusrcb 01
47 assert aluop 00
48 assert pcchange 1
49 assert pcsource 00
50
51 | state 2
52 c
53 assert memread 1
54 assert alusrca 0
55 assert iord 0
56 assert irwrite 0010
57 assert alusrcb 01
58 assert aluop 00
59 assert pcchange 1
60 assert pcsource 00
61
62 | state 3
63 c
64 assert memread 1
65 assert alusrca 0
66 assert iord 0
67 assert irwrite 0001
68 assert alusrcb 01
69 assert aluop 00
70 assert pcchange 1
71 assert pcsource 00
72
73 | state 4
74 c
75 assert alusrca 0
76 assert alusrcb 11
77 assert aluop 00
78
79 | state 5
80 c
81 assert alusrca 1
82 assert alusrcb 10
83 assert aluop 00
84
85 | state 6
86 c
87 assert memread 1
```



```
88 assert iord 1
89
90 | state 7
91 c
92 assert regdst 0
93 assert regwrite 1
94 assert memtoreg 1
95
96 | ***** test sb
97 | state 0
98 set op 011000
99 c
100 assert memread 1
101 assert alusrca 0
102 assert iord 0
103 assert irwrite 1000
104 assert alusrcb 01
105 assert aluop 00
106 assert pcchange 1
107 assert pcsource 00
108
109 | state 1
110 c
111 assert memread 1
112 assert alusrca 0
113 assert iord 0
114 assert irwrite 0100
115 assert alusrcb 01
116 assert aluop 00
117 assert pcchange 1
118 assert pcsource 00
119
120 | state 2
121 c
122 assert memread 1
123 assert alusrca 0
124 assert iord 0
125 assert irwrite 0010
126 assert alusrcb 01
127 assert aluop 00
128 assert pcchange 1
129 assert pcsource 00
130
131 | state 3
132 c
133 assert memread 1
134 assert alusrca 0
135 assert iord 0
136 assert irwrite 0001
137 assert alusrcb 01
138 assert aluop 00
139 assert pcchange 1
140 assert pcsource 00
141
142 | state 4
143 c
144 assert alusrca 0
145 assert alusrcb 11
```

```
146 assert aluop 00
147
148 | state 5
149 c
150 assert alusrca 1
151 assert alusrcb 10
152 assert aluop 00
153
154 | state 8
155 c
156 assert memwrite 1
157 assert iord 1
158
159 | ***** test R-type instructions
160 | state 0
161 set op 000000
162 c
163 assert memread 1
164 assert alusrca 0
165 assert iord 0
166 assert irwrite 1000
167 assert alusrcb 01
168 assert aluop 00
169 assert pcchange 1
170 assert pcsource 00
171
172 | state 1
173 c
174 assert memread 1
175 assert alusrca 0
176 assert iord 0
177 assert irwrite 0100
178 assert alusrcb 01
179 assert aluop 00
180 assert pcchange 1
181 assert pcsource 00
182
183 | state 2
184 c
185 assert memread 1
186 assert alusrca 0
187 assert iord 0
188 assert irwrite 0010
189 assert alusrcb 01
190 assert aluop 00
191 assert pcchange 1
192 assert pcsource 00
193
194 | state 3
195 c
196 assert memread 1
197 assert alusrca 0
198 assert iord 0
199 assert irwrite 0001
200 assert alusrcb 01
201 assert aluop 00
202 assert pcchange 1
203 assert pcsource 00
```

```
204
205 | state 4
206 c
207 assert alusrca 0
208 assert alusrcb 11
209 assert aluop 00
210
211 | state 9
212 c
213 assert alusrca 1
214 assert alusrcb 00
215 assert aluop 10
216
217 | state 10
218 c
219 assert regdst 1
220 assert regwrite 1
221 assert memtoreg 0
222
223 | ***** test beq
224 | state 0
225 set op 000100
226 c
227 assert memread 1
228 assert alusrca 0
229 assert iord 0
230 assert irwrite 1000
231 assert alusrcb 01
232 assert aluop 00
233 assert pcchange 1
234 assert pcsource 00
235
236 | state 1
237 c
238 assert memread 1
239 assert alusrca 0
240 assert iord 0
241 assert irwrite 0100
242 assert alusrcb 01
243 assert aluop 00
244 assert pcchange 1
245 assert pcsource 00
246
247 | state 2
248 c
249 assert memread 1
250 assert alusrca 0
251 assert iord 0
252 assert irwrite 0010
253 assert alusrcb 01
254 assert aluop 00
255 assert pcchange 1
256 assert pcsource 00
257
258 | state 3
259 c
260 assert memread 1
261 assert alusrca 0
```

```
262 assert iord 0
263 assert irwrite 0001
264 assert alusrcb 01
265 assert aluop 00
266 assert pcchange 1
267 assert pcsource 00
268
269 | state 4
270 c
271 assert alusrca 0
272 assert alusrcb 11
273 assert aluop 00
274
275 | state 11
276 c
277 assert alusrca 1
278 assert alusrcb 00
279 assert aluop 01
280 assert pcsource 01
281 assert pcchange 0
282
283 | check that zero works in state 11
284 h zero
285 s
286 assert pcchange 1
287
288 | ***** test j
289 | state 0
290 set op 000010
291 c
292 assert memread 1
293 assert alusrca 0
294 assert iord 0
295 assert irwrite 1000
296 assert alusrcb 01
297 assert aluop 00
298 assert pcchange 1
299 assert pcsource 00
300
301 | state 1
302 c
303 assert memread 1
304 assert alusrca 0
305 assert iord 0
306 assert irwrite 0100
307 assert alusrcb 01
308 assert aluop 00
309 assert pcchange 1
310 assert pcsource 00
311
312 | state 2
313 c
314 assert memread 1
315 assert alusrca 0
316 assert iord 0
317 assert irwrite 0010
318 assert alusrcb 01
319 assert aluop 00
```

```
320 assert pcchange 1
321 assert pcsource 00
322
323 | state 3
324 c
325 assert memread 1
326 assert alusrca 0
327 assert iord 0
328 assert irwrite 0001
329 assert alusrcb 01
330 assert aluop 00
331 assert pcchange 1
332 assert pcsource 00
333
334 | state 4
335 c
336 assert alusrca 0
337 assert alusrcb 11
338 assert aluop 00
339
340 | state 12
341 c
342 assert pcsource 10
343 assert pcchange 1
344
345 | ***** test addi
346 | state 0
347 set op 001000
348 c
349 assert memread 1
350 assert alusrca 0
351 assert iord 0
352 assert irwrite 1000
353 assert alusrcb 01
354 assert aluop 00
355 assert pcchange 1
356 assert pcsource 00
357
358 | state 1
359 c
360 assert memread 1
361 assert alusrca 0
362 assert iord 0
363 assert irwrite 0100
364 assert alusrcb 01
365 assert aluop 00
366 assert pcchange 1
367 assert pcsource 00
368
369 | state 2
370 c
371 assert memread 1
372 assert alusrca 0
373 assert iord 0
374 assert irwrite 0010
375 assert alusrcb 01
376 assert aluop 00
377 assert pcchange 1
```

```
378 assert pcsource 00
379
380 | state 3
381 c
382 assert memread 1
383 assert alusrca 0
384 assert iord 0
385 assert irwrite 0001
386 assert alusrca 01
387 assert aluop 00
388 assert pcchange 1
389 assert pcsource 00
390
391 | state 4
392 c
393 assert alusrca 0
394 assert alusrca 11
395 assert aluop 00
396
397 | state 13
398 c
399 assert alusrca 1
400 assert alusrca 10
401 assert aluop 00
402
403 | state 14
404 c
405 assert regdst 0
406 assert regwrite 1
407 assert memtoreg 0
408
409 | *** all MIPS instructions tested
```