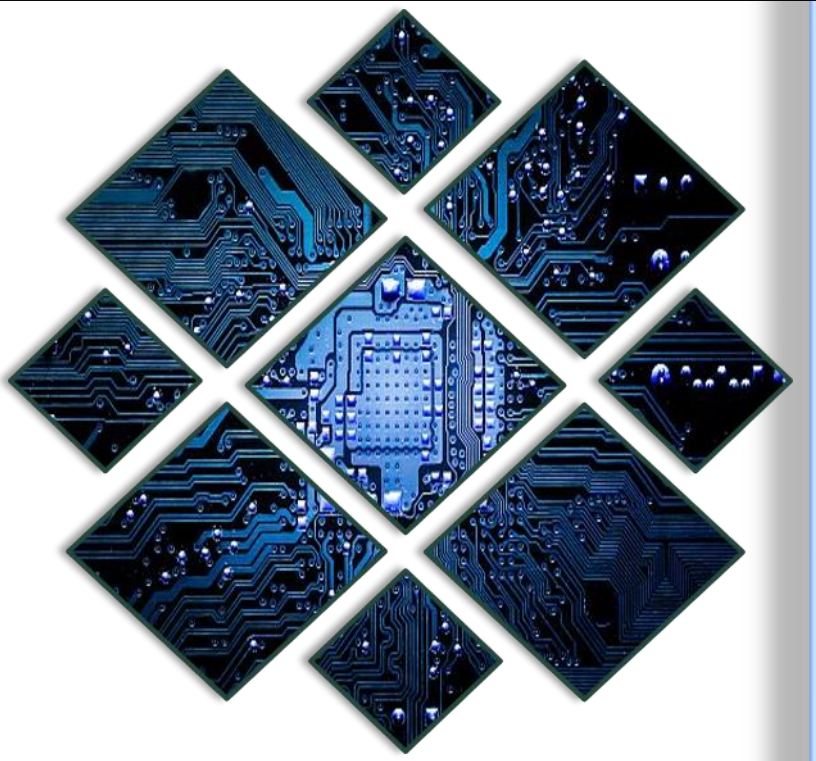
VLSI

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

```
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)
```

NCC

```
================================ 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

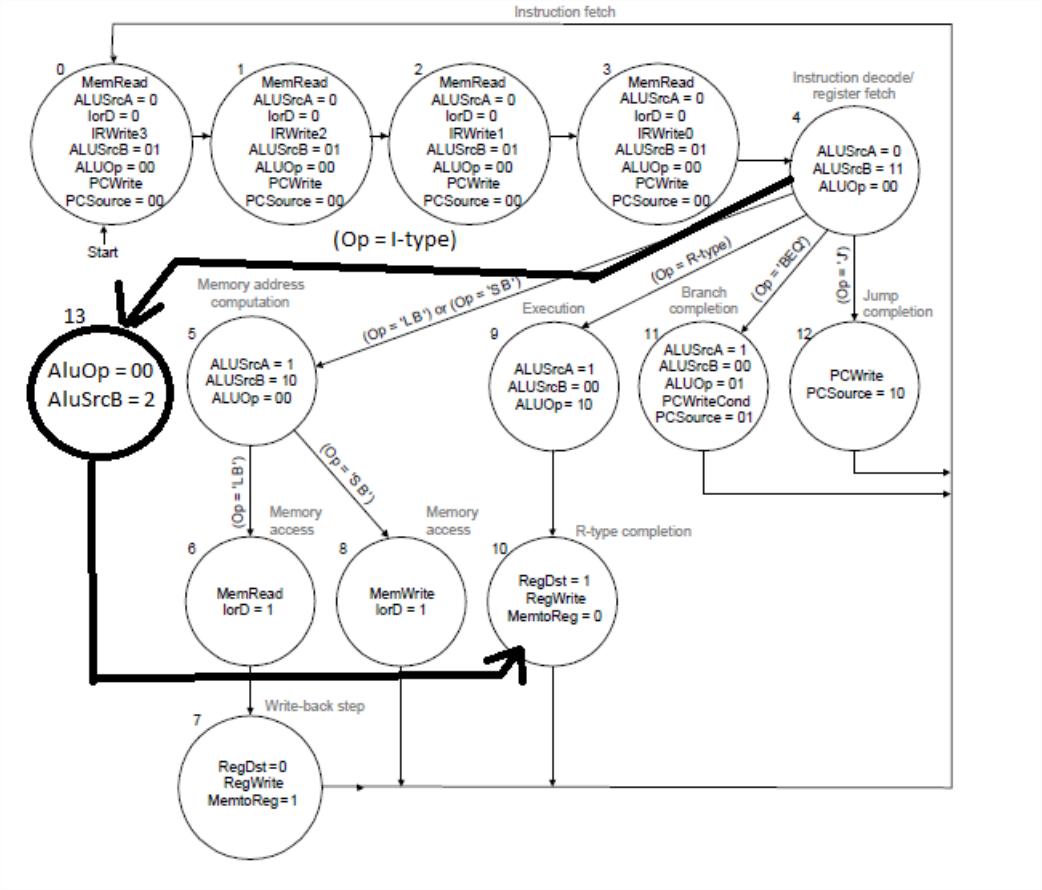
```
================================ 8 ================================
Found 233 networks, allocated 2820 bytes for network numbering (took 0.003 seconds)
0 errors found (took 0.007 seconds)
-------------------------------- 9 --------------------------------
```

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)
```

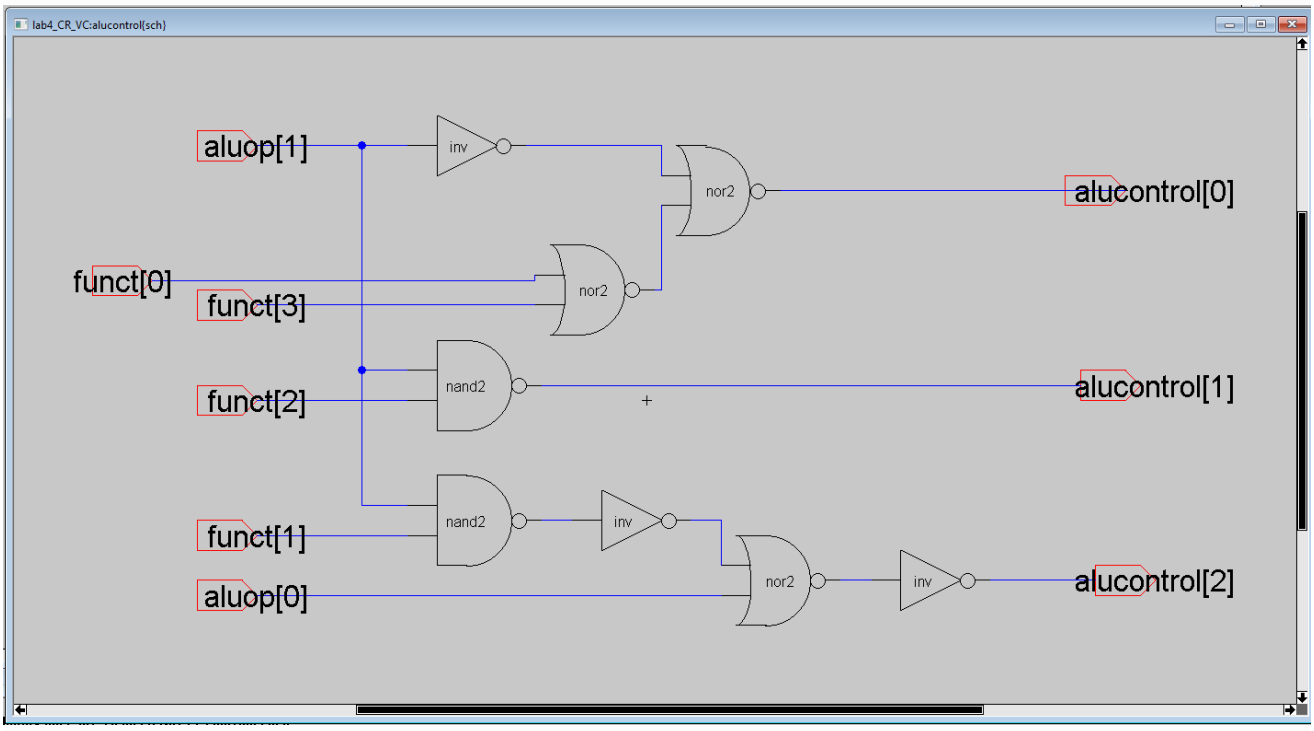
```
================================ 10 ================================
Comparing facet std_nor3{sch} with facet std_nor3{lay}
- Checking facets recursively; Ignoring Power and Ground nets
- Parallel components not merged; Series transistors not merged
- Checking export names and component sizes
- No 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

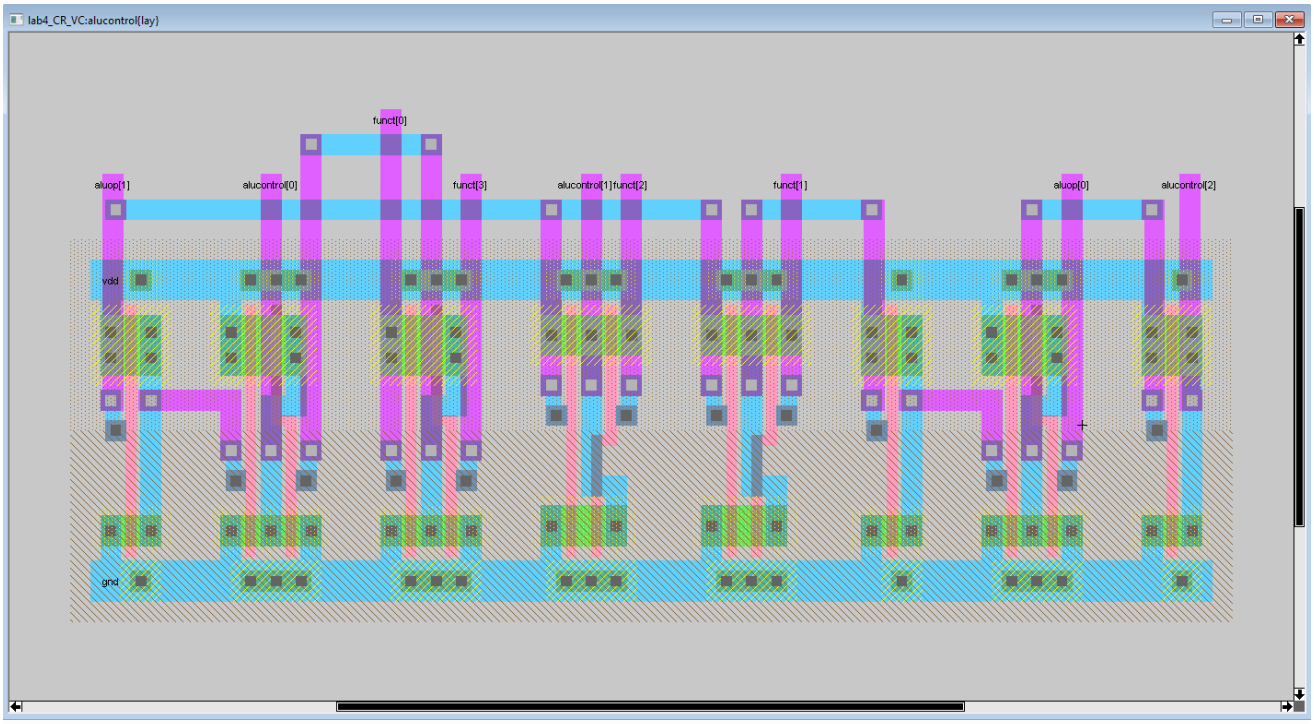


4. Printout alucontrol

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.

```
| alucontrol.cmd
| Written 2/12/02 David_Harris@hmc.edu
|
| Simulate the alucontrol block.
| only consider legal instructions
| Note: this file is incomplete

vector aluop aluop[1] aluop[0]
vector funct funct[3] funct[2] funct[1] funct[0]
vector alucontrol alucontrol[2] alucontrol[1] alucontrol[0]

| test always adding
set aluop 00
s 2
assert alucontrol 010

| test always subtracting
set aluop 01
s 2
assert alucontrol 110

| test funct ADD
set aluop 10
set funct 0000
s 2
assert alucontrol 010

| add code here to test SUB, AND, OR, SLT

| test always SUB
set aluop 10
```

```
set funct 0010
s 2
assert alucontrol 110

| test always AND
set aluop 10
set funct 0100
s 2
assert alucontrol 000

| test always OR
set aluop 10
set funct 0101
s 2
assert alucontrol 001

| test always SLT
set aluop 10
set funct 1010
s 2
assert alucontrol 111
```

6. Controller.cmd

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

```
| controller.cmd
| Written 2/12/02 David_Harris@hmc.edu
|
| Simulate the controller for lab 4.

| define vectors for convenience
vector op op5 op4 op3 op2 op1 op0
vector irwrite irwrite3 irwrite2 irwrite1 irwrite0
vector aluop aluop1 aluop0
vector alusrcb alusrcb1 alusrcb0
vector pcsource pcsource1 pcsource0

| define a 2-phase nonoverlapping clock
| in which ph1 goes high then low
| then ph2 goes high then low
| with a time of 2 ns for each step (8 ns cycle time)
clock ph1 1 0 0 0
clock ph2 0 0 1 0
stepsize 2

| reset the system by setting reset high and clocking twice
h reset
l zero
c
l reset

| ***** test lb
| state 0
set op 010000
```

```
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 1000
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 1
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0100
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 2
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0010
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 3
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0001
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 4
c
assert alusrca 0
assert alusrcb 11
assert aluop 00

| state 5
c
assert alusrca 1
assert alusrcb 10
assert aluop 00

| state 6
c
assert memread 1
```

```
assert iord 1

| state 7
c
assert regdst 0
assert regwrite 1
assert memtoreg 1

| ***** test sb
| state 0
set op 011000
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 1000
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 1
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0100
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 2
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0010
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 3
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0001
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 4
c
assert alusrca 0
assert alusrcb 11
```

```
assert aluop 00

| state 5
c
assert alusrca 1
assert alusrcb 10
assert aluop 00

| state 8
c
assert memwrite 1
assert iord 1

| ***** test R-type instructions
| state 0
set op 000000
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 1000
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 1
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0100
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 2
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0010
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 3
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0001
assert alusrcb 01
assert aluop 00
assert pcchange 1
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
```

```
assert iord 0
assert irwrite 0001
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 4
c
assert alusrca 0
assert alusrcb 11
assert aluop 00

| state 11
c
assert alusrca 1
assert alusrcb 00
assert aluop 01
assert pcsource 01
assert pcchange 0

| check that zero works in state 11
h zero
s
assert pcchange 1

| ***** test j
| state 0
set op 000010
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 1000
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 1
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0100
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 2
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0010
assert alusrcb 01
assert aluop 00
```

```
assert pcchange 1
assert pcsource 00

| state 3
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0001
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 4
c
assert alusrca 0
assert alusrcb 11
assert aluop 00

| state 12
c
assert pcsource 10
assert pcchange 1

| ***** test addi
| state 0
set op 001000
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 1000
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 1
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0100
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 2
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0010
assert alusrcb 01
assert aluop 00
assert pcchange 1
```

```
assert pcsource 00

| state 3
c
assert memread 1
assert alusrca 0
assert iord 0
assert irwrite 0001
assert alusrcb 01
assert aluop 00
assert pcchange 1
assert pcsource 00

| state 4
c
assert alusrca 0
assert alusrcb 11
assert aluop 00

| state 13
c
assert alusrca 1
assert alusrcb 10
assert aluop 00

| state 14
c
assert regdst 0
assert regwrite 1
assert memtoreg 0

| *** all MIPS instructions tested
```