CprE 381: Computer Organization and Assembly-Level Programming

Project Part 1 Report

Team Members: Hunter Northern

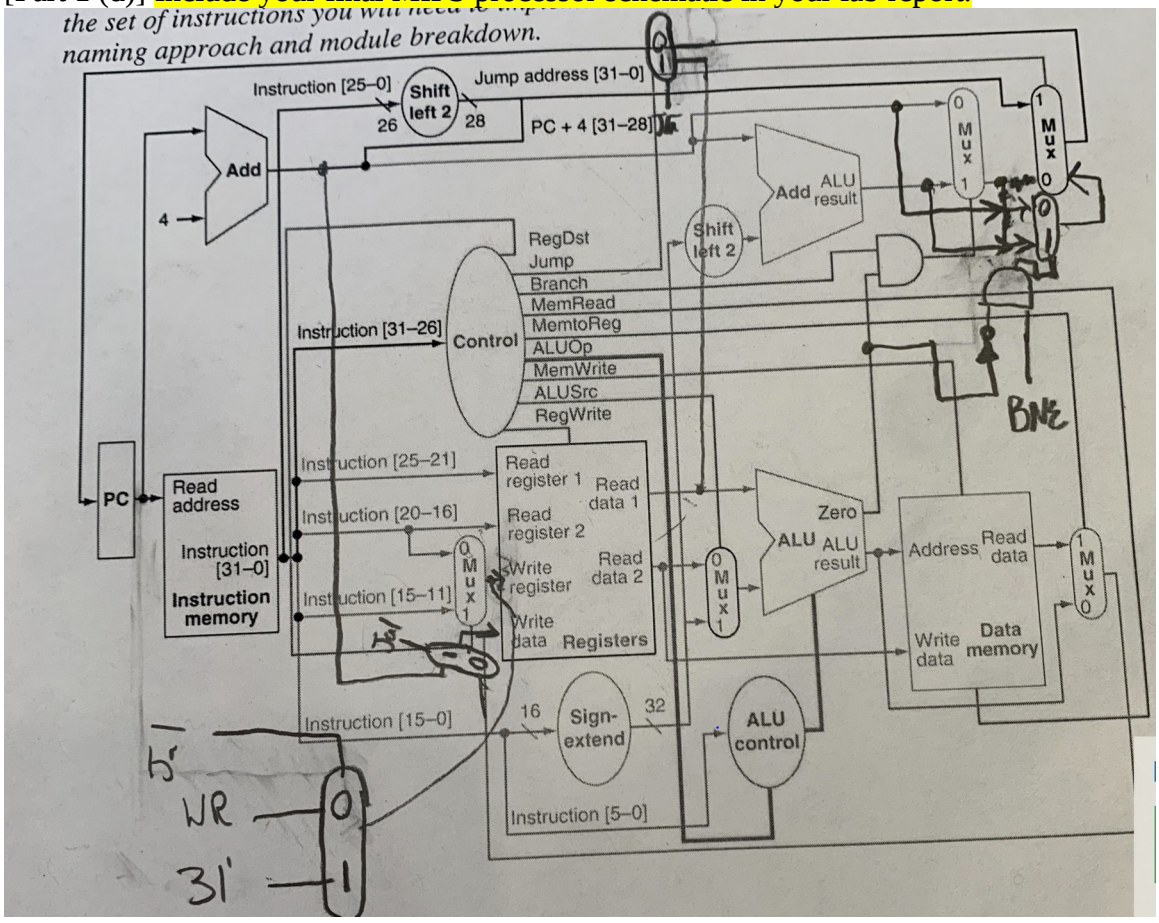
Bridget Schmitt

Jeremy Noessen

Project Teams Group #: _____

Refer to the highlighted language in the project 1 instruction for the context of the following questions.

[Part 1 (d)] Include your final MIPS processor schematic in your lab report.

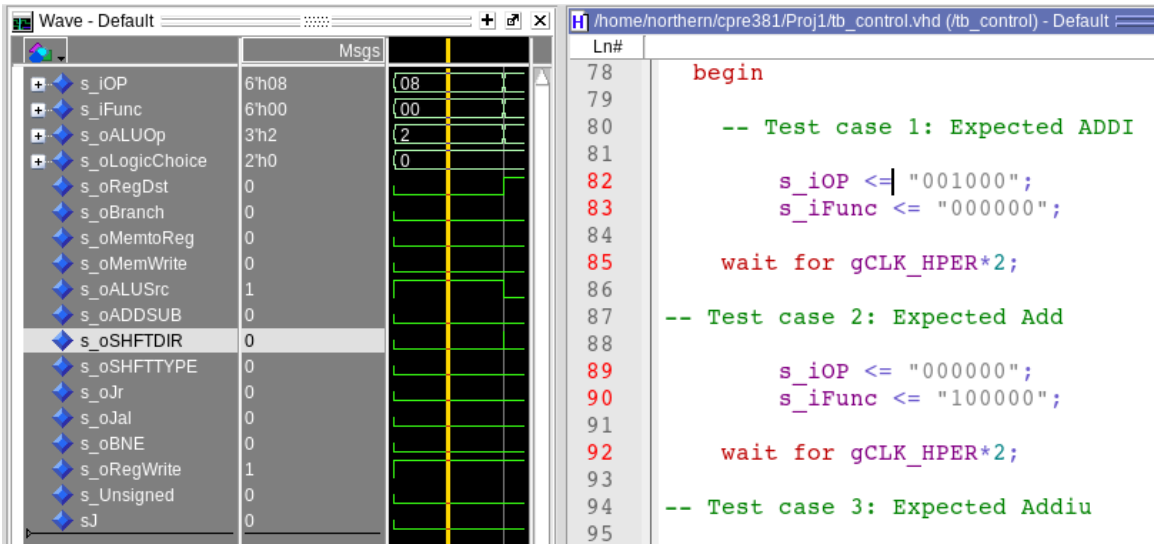


[Part 2 (a.i)] Create a spreadsheet detailing the list of *M* instructions to be supported in your project alongside their binary opcodes and funct fields, if applicable. Create a separate column for each binary bit. Inside this spreadsheet, create a new column for the *N* control signals needed by your datapath implementation. The end result should be an *N*M* table where each row corresponds to the output of the control logic module for a given instruction.

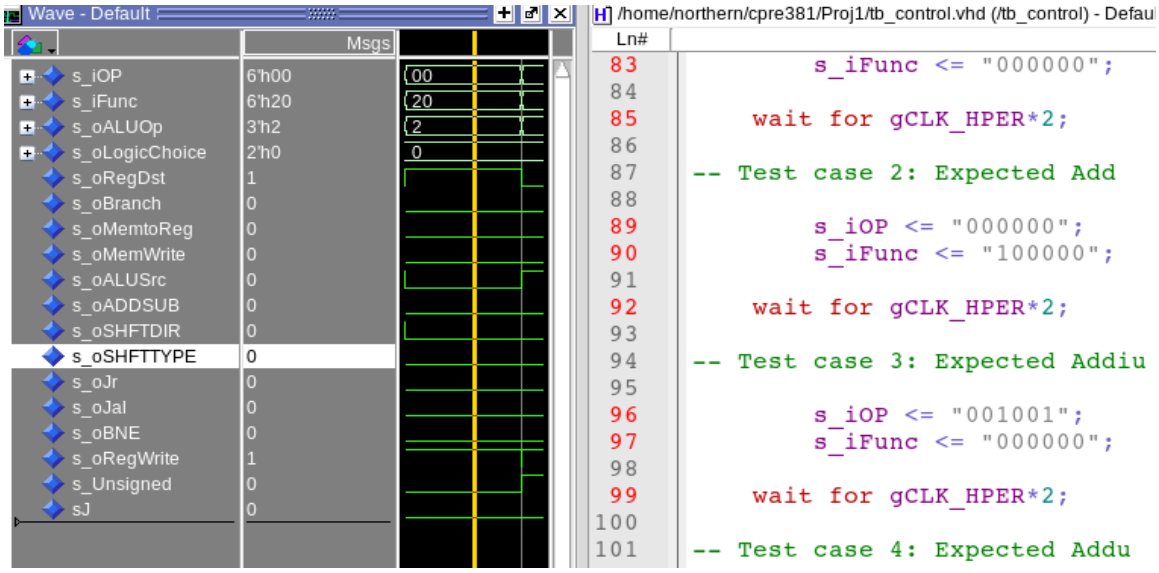
Included as Proj1_control_signals.xlsx

[Part 2 (a.ii)] Implement the control logic module using whatever method and coding style you prefer. Create a testbench to test this module individually, and show that your output matches the expected control signals from problem 1(a).

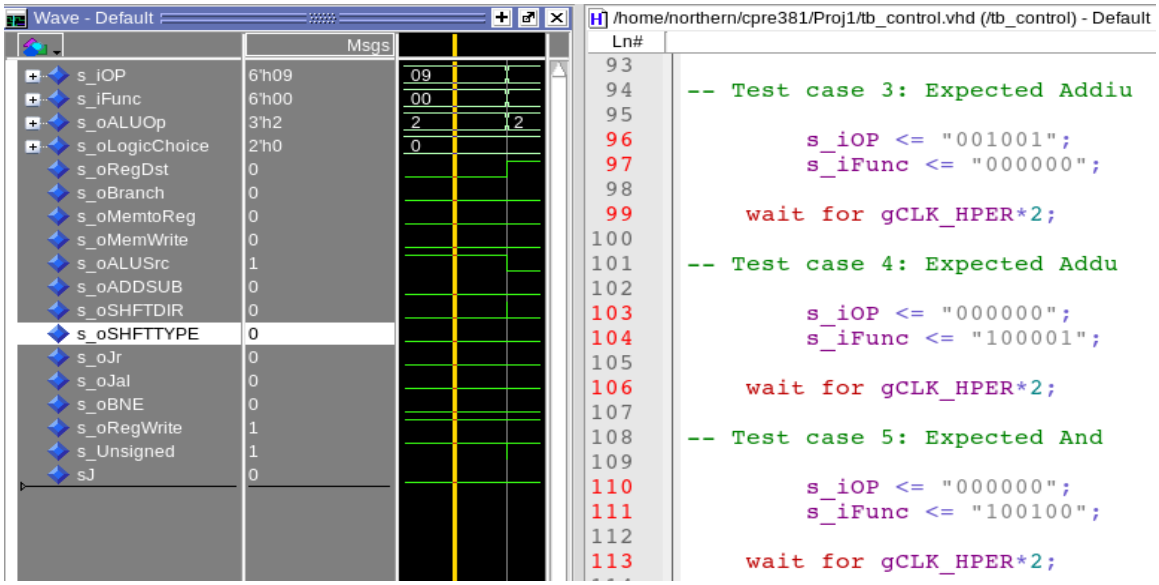
Addi:



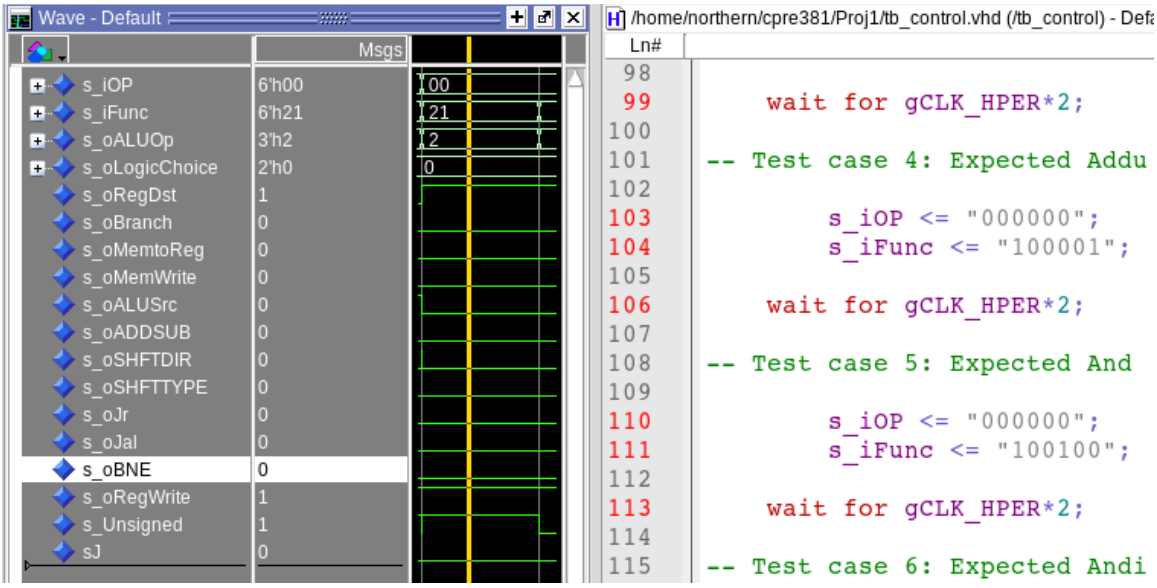
Add:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h20
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
00
20
2
0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defaul
Ln#
83 s_iFunc <= "000000";
84
85 wait for gCLK_HPER*2;
86
87 -- Test case 2: Expected Add
88
89 s_iOP <= "000000";
90 s_iFunc <= "100000";
91
92 wait for gCLK_HPER*2;
93
94 -- Test case 3: Expected Addiu
95
96 s_iOP <= "001001";
97 s_iFunc <= "000000";
98
99 wait for gCLK_HPER*2;
100
101 -- Test case 4: Expected Addu

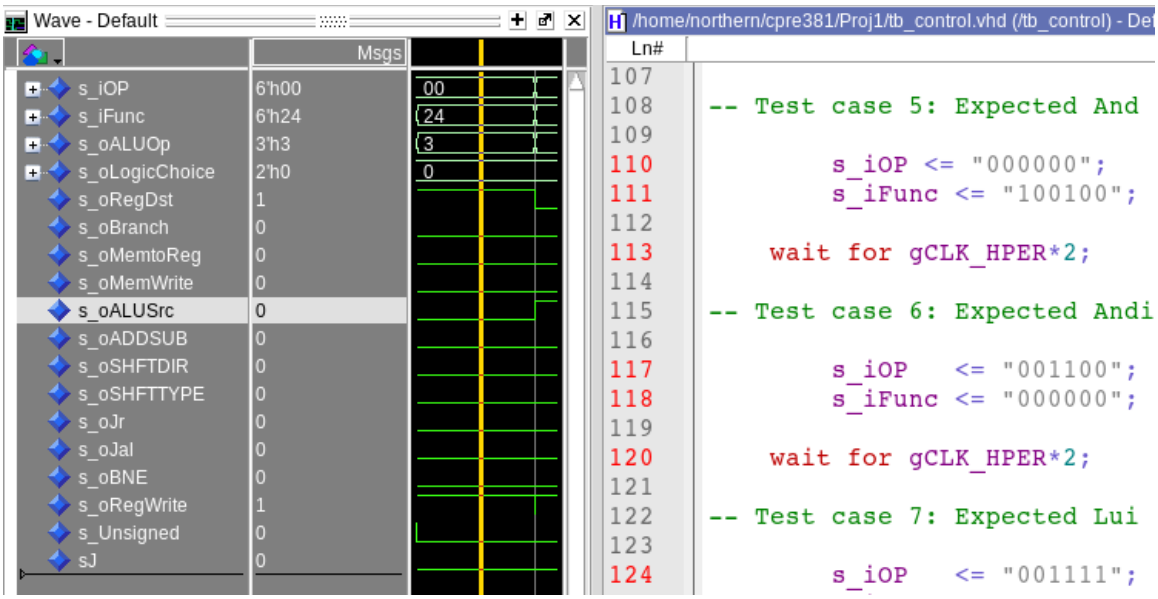
Addiu:


Wave - Default
Msgs
s_iOP 6'h09
s_iFunc 6'h00
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 1
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 1
sJ 0
09
00
2
0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Default
Ln#
93
94 -- Test case 3: Expected Addiu
95
96 s_iOP <= "001001";
97 s_iFunc <= "000000";
98
99 wait for gCLK_HPER*2;
100
101 -- Test case 4: Expected Addu
102
103 s_iOP <= "000000";
104 s_iFunc <= "100001";
105
106 wait for gCLK_HPER*2;
107
108 -- Test case 5: Expected And
109
110 s_iOP <= "000000";
111 s_iFunc <= "100100";
112
113 wait for gCLK_HPER*2;

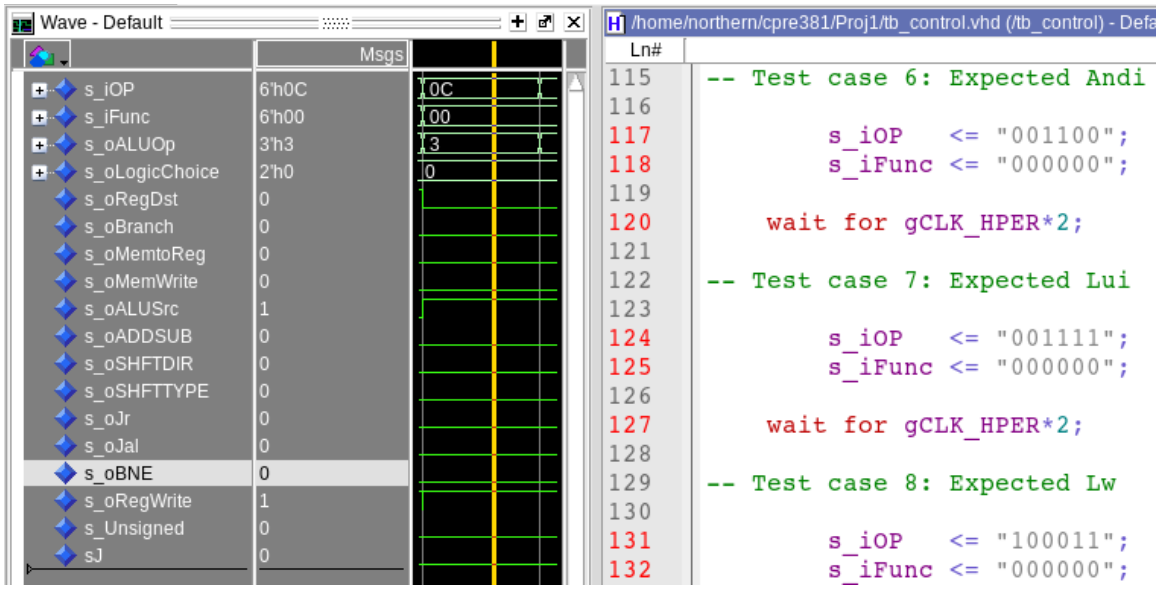
Addu:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h21
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 1
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
98
99 wait for gCLK_HPER*2;
100
101 -- Test case 4: Expected Addu
102
103 s_iOP <= "000000";
104 s_iFunc <= "100001";
105
106 wait for gCLK_HPER*2;
107
108 -- Test case 5: Expected And
109
110 s_iOP <= "000000";
111 s_iFunc <= "100100";
112
113 wait for gCLK_HPER*2;
114
115 -- Test case 6: Expected Andi

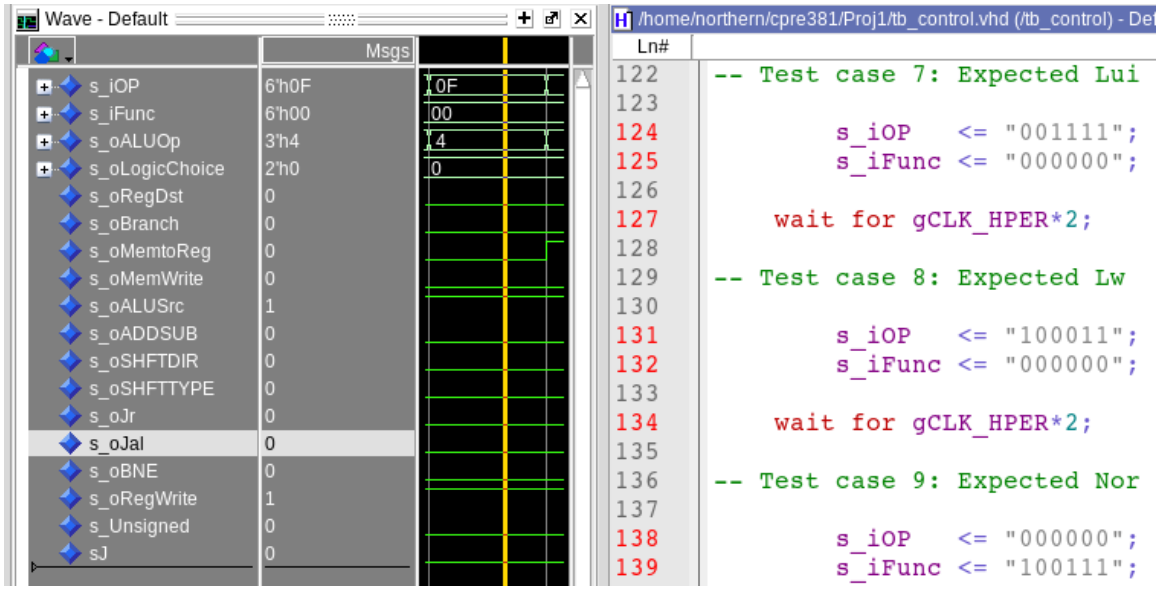
And:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h24
s_oALUOp 3'h3
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Def
Ln#
107
108 -- Test case 5: Expected And
109
110 s_iOP <= "000000";
111 s_iFunc <= "100100";
112
113 wait for gCLK_HPER*2;
114
115 -- Test case 6: Expected Andi
116
117 s_iOP <= "001100";
118 s_iFunc <= "000000";
119
120 wait for gCLK_HPER*2;
121
122 -- Test case 7: Expected Lui
123
124 s_iOP <= "001111";

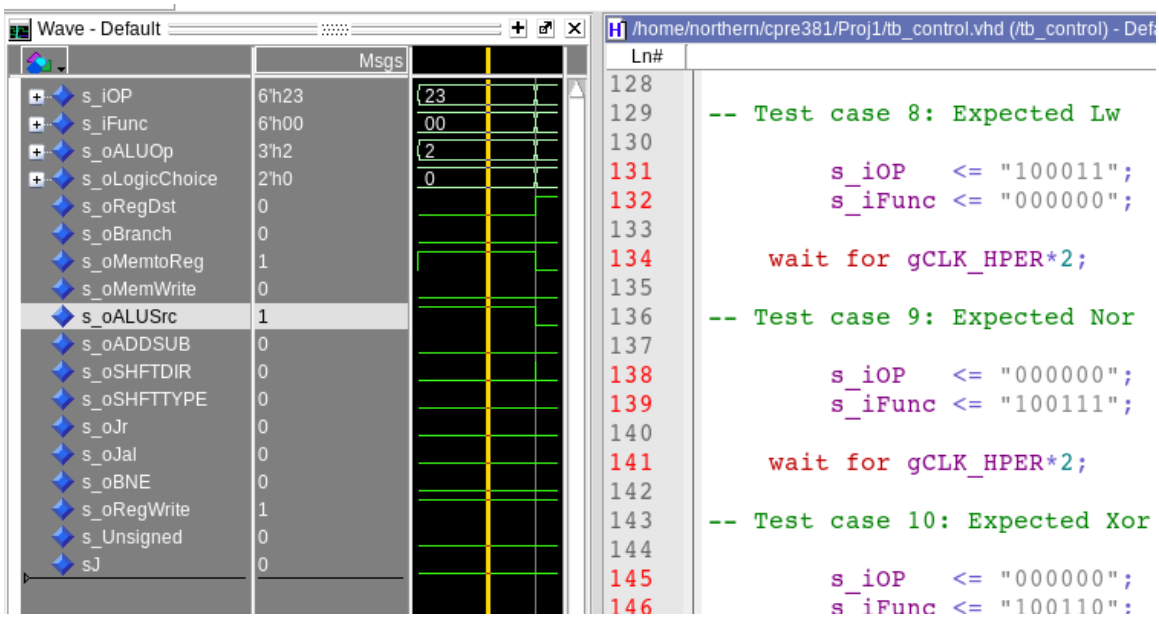
Andi:


Wave - Default
Msgs
s_iOP 6'h0C
s_iFunc 6'h00
s_oALUOp 3'h3
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 1
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
115 -- Test case 6: Expected Andi
116
117 s_iOP <= "001100";
118 s_iFunc <= "000000";
119
120 wait for gCLK_HPER*2;
121
122 -- Test case 7: Expected Lui
123
124 s_iOP <= "001111";
125 s_iFunc <= "000000";
126
127 wait for gCLK_HPER*2;
128
129 -- Test case 8: Expected Lw
130
131 s_iOP <= "100011";
132 s_iFunc <= "000000";

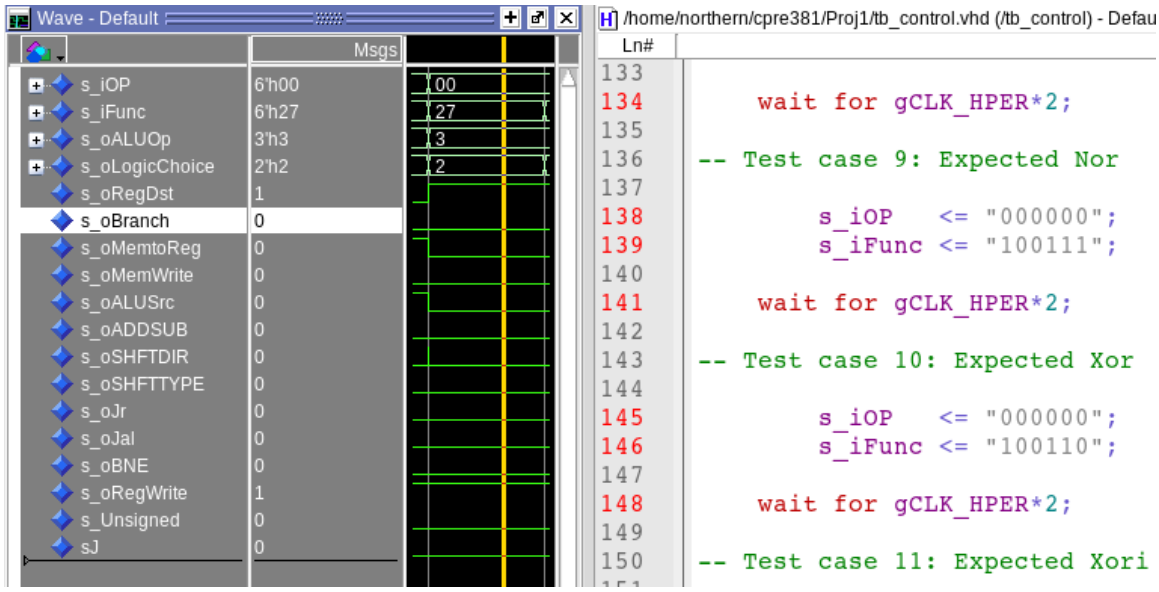
Lui:


Wave - Default
Msgs
s_iOP 6'h0F
s_iFunc 6'h00
s_oALUOp 3'h4
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 1
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Def
Ln#
122 -- Test case 7: Expected Lui
123
124 s_iOP <= "001111";
125 s_iFunc <= "000000";
126
127 wait for gCLK_HPER*2;
128
129 -- Test case 8: Expected Lw
130
131 s_iOP <= "100011";
132 s_iFunc <= "000000";
133
134 wait for gCLK_HPER*2;
135
136 -- Test case 9: Expected Nor
137
138 s_iOP <= "000000";
139 s_iFunc <= "100111";

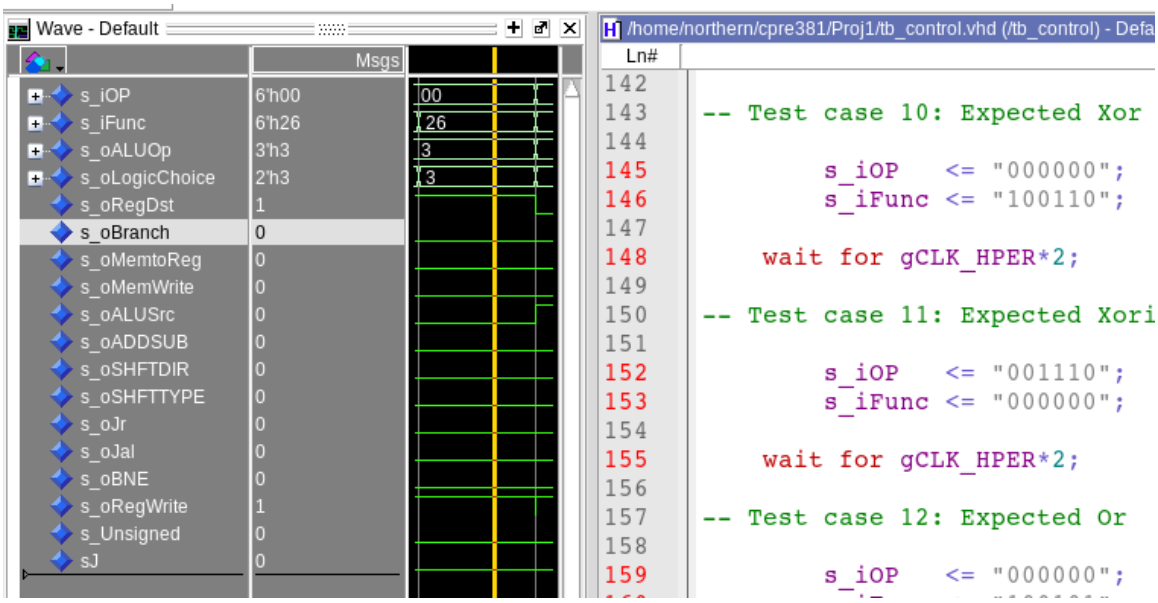
Lw:


Wave - Default
Msgs
s_iOP 6'h23
s_iFunc 6'h00
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 1
s_oMemWrite 0
s_oALUSrc 1
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Def
Ln#
128
129 -- Test case 8: Expected Lw
130
131 s_iOP <= "100011";
132 s_iFunc <= "000000";
133
134 wait for gCLK_HPER*2;
135
136 -- Test case 9: Expected Nor
137
138 s_iOP <= "000000";
139 s_iFunc <= "100111";
140
141 wait for gCLK_HPER*2;
142
143 -- Test case 10: Expected Xor
144
145 s_iOP <= "000000";

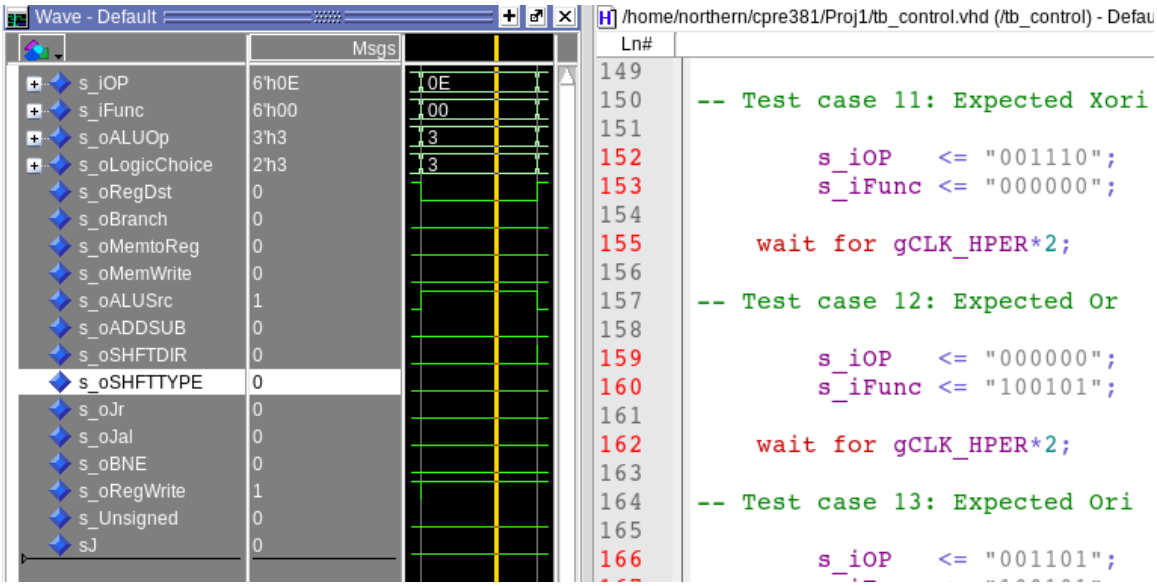
Nor:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h27
s_oALUOp 3'h3
s_oLogicChoice 2'h2
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defau
Ln#
133
134 wait for gCLK_HPER*2;
135
136 -- Test case 9: Expected Nor
137
138 s_iOP <= "000000";
139 s_iFunc <= "100111";
140
141 wait for gCLK_HPER*2;
142
143 -- Test case 10: Expected Xor
144
145 s_iOP <= "000000";
146 s_iFunc <= "100110";
147
148 wait for gCLK_HPER*2;
149
150 -- Test case 11: Expected Xori

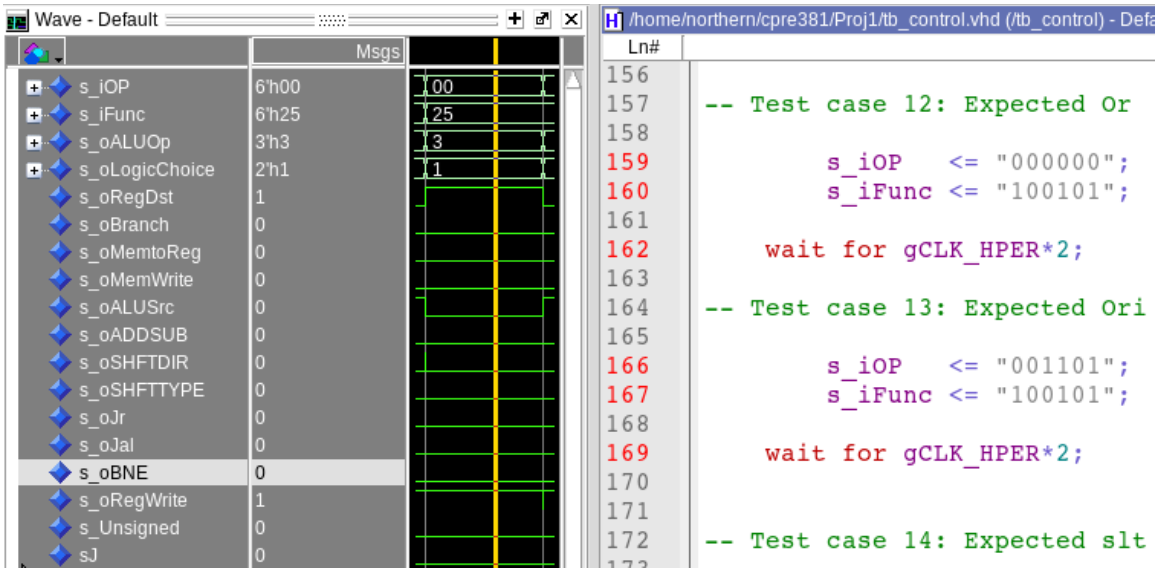
Xor:


Wave - Default
Msgs
s_iOP
s_iFunc
s_oALUOp
s_oLogicChoice
s_oRegDst
s_oBranch
s_oMemtoReg
s_oMemWrite
s_oALUSrc
s_oADDSUB
s_oSHFTDIR
s_oSHFTTYPE
s_oJr
s_oJal
s_oBNE
s_oRegWrite
s_Unsigned
sJ
6'h00
6'h26
3'h3
2'h3
1
0
00
26
3
3
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
142
143 -- Test case 10: Expected Xor
144
145 s_iOP <= "000000";
146 s_iFunc <= "100110";
147
148 wait for gCLK_HPER*2;
149
150 -- Test case 11: Expected Xori
151
152 s_iOP <= "001110";
153 s_iFunc <= "000000";
154
155 wait for gCLK_HPER*2;
156
157 -- Test case 12: Expected Or
158
159 s_iOP <= "000000";

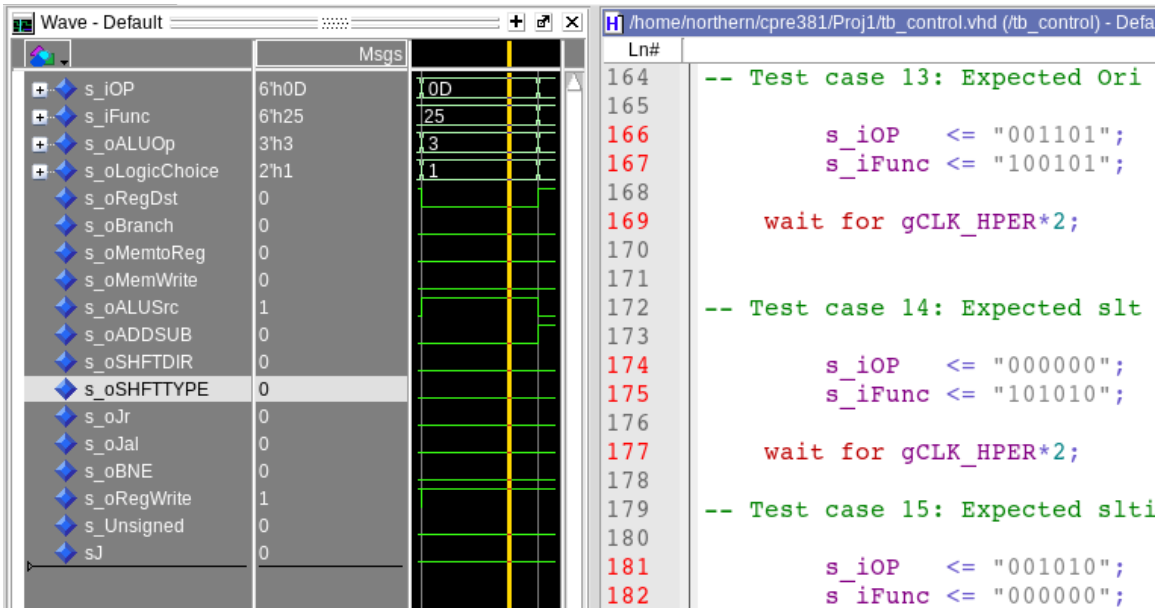
Xori:


Wave - Default
Msgs
s_iOP
s_iFunc
s_oALUOp
s_oLogicChoice
s_oRegDst
s_oBranch
s_oMemtoReg
s_oMemWrite
s_oALUSrc
s_oADDSUB
s_oSHFTDIR
s_oSHFTTYPE
s_oJr
s_oJal
s_oBNE
s_oRegWrite
s_Unsigned
sJ
6'h0E
6'h00
3'h3
2'h3
0
1
0E
00
3
3
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defau
Ln#
149
150 -- Test case 11: Expected Xori
151
152 s_iOP <= "001110";
153 s_iFunc <= "000000";
154
155 wait for gCLK_HPER*2;
156
157 -- Test case 12: Expected Or
158
159 s_iOP <= "000000";
160 s_iFunc <= "100101";
161
162 wait for gCLK_HPER*2;
163
164 -- Test case 13: Expected Ori
165
166 s_iOP <= "001101";

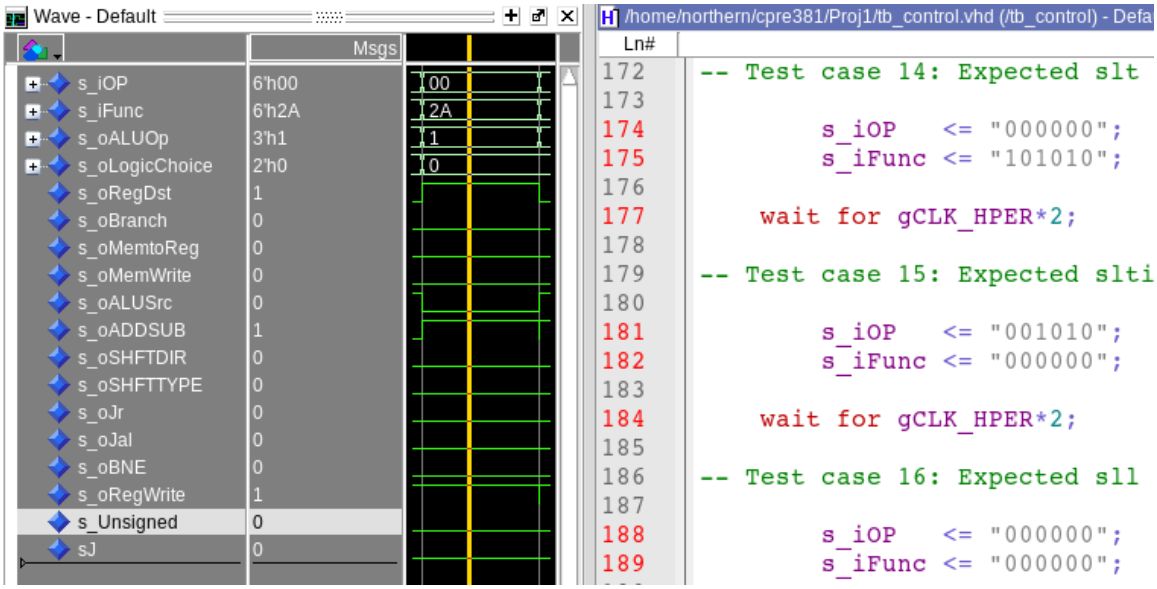
Or:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h25
s_oALUOp 3'h3
s_oLogicChoice 2'h1
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
00
25
3
1
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
156
157 -- Test case 12: Expected Or
158
159 s_iOP <= "000000";
160 s_iFunc <= "100101";
161
162 wait for gCLK_HPER*2;
163
164 -- Test case 13: Expected Ori
165
166 s_iOP <= "001101";
167 s_iFunc <= "100101";
168
169 wait for gCLK_HPER*2;
170
171
172 -- Test case 14: Expected slt

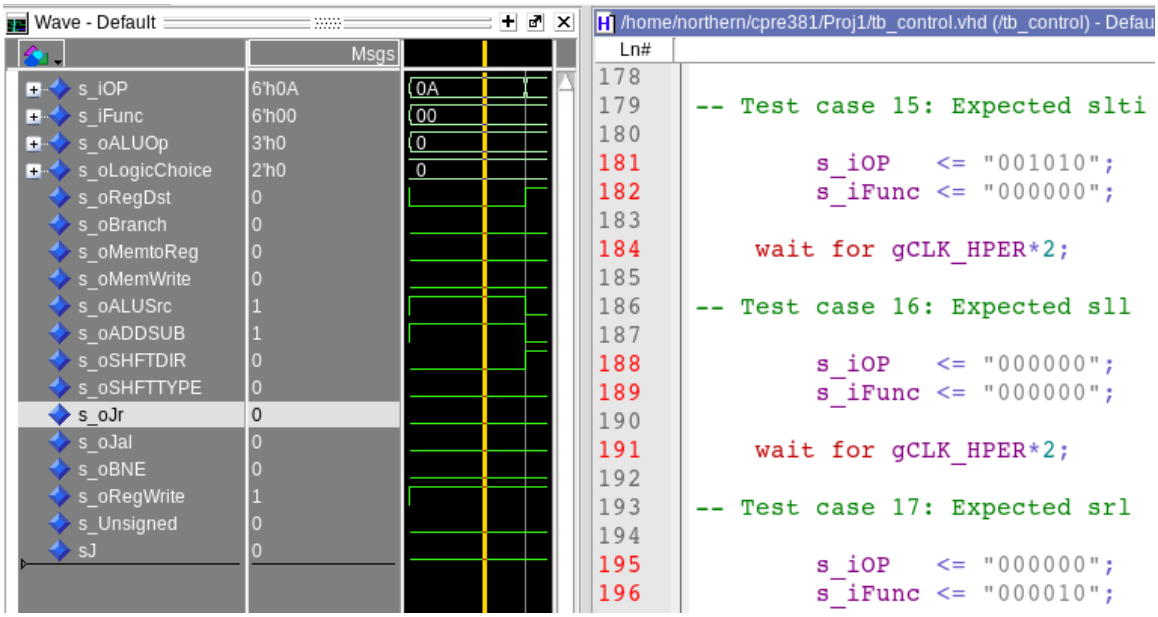
Ori:


Wave - Default
Msgs
s_iOP 6'h0D
s_iFunc 6'h25
s_oALUOp 3'h3
s_oLogicChoice 2'h1
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 1
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
0D
25
3
1
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
164 -- Test case 13: Expected Ori
165
166 s_iOP <= "001101";
167 s_iFunc <= "100101";
168
169 wait for gCLK_HPER*2;
170
171
172 -- Test case 14: Expected slt
173
174 s_iOP <= "000000";
175 s_iFunc <= "101010";
176
177 wait for gCLK_HPER*2;
178
179 -- Test case 15: Expected slti
180
181 s_iOP <= "001010";
182 s_iFunc <= "000000";

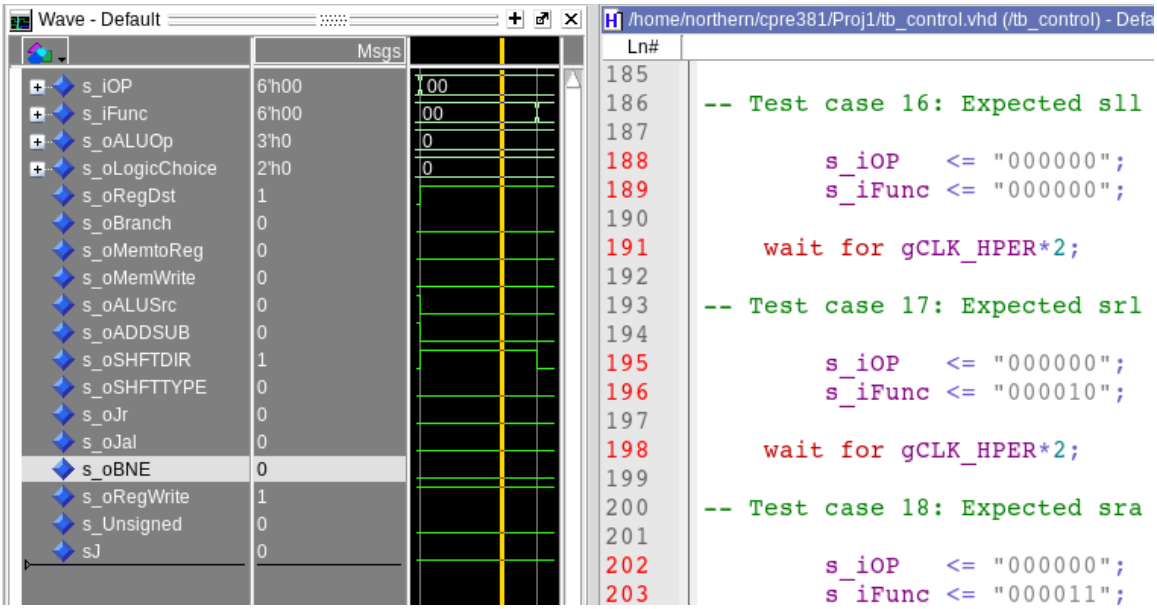
slt:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h2A
s_oALUOp 3'h1
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 1
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
00
2A
1
0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
172 -- Test case 14: Expected slt
173
174 s_iOP <= "000000";
175 s_iFunc <= "101010";
176
177 wait for gCLK_HPER*2;
178
179 -- Test case 15: Expected slti
180
181 s_iOP <= "001010";
182 s_iFunc <= "000000";
183
184 wait for gCLK_HPER*2;
185
186 -- Test case 16: Expected sll
187
188 s_iOP <= "000000";
189 s_iFunc <= "000000";

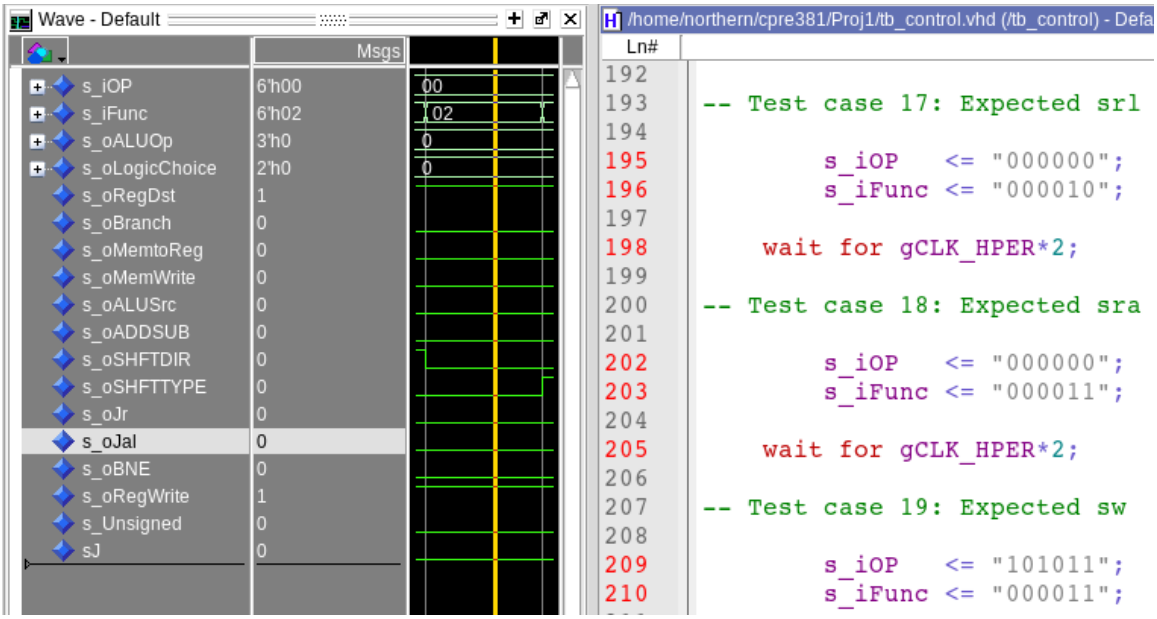
slti:


Wave - Default
Msgs
s_iOP 6'h0A
s_iFunc 6'h00
s_oALUOp 3'h0
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 1
s_oADDSUB 1
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
0A
00
0
0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defau
Ln#
178
179 -- Test case 15: Expected slti
180
181 s_iOP <= "001010";
182 s_iFunc <= "000000";
183
184 wait for gCLK_HPER*2;
185
186 -- Test case 16: Expected sll
187
188 s_iOP <= "000000";
189 s_iFunc <= "000000";
190
191 wait for gCLK_HPER*2;
192
193 -- Test case 17: Expected srl
194
195 s_iOP <= "000000";
196 s_iFunc <= "000010";

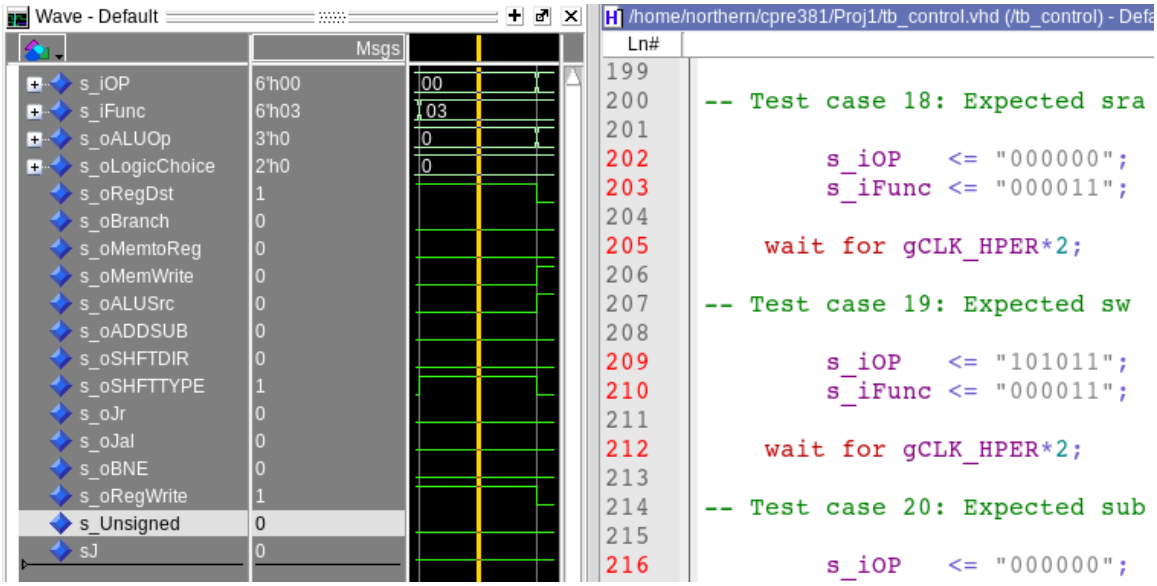
sll:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h00
s_oALUOp 3'h0
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 1
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
185
186 -- Test case 16: Expected sll
187
188 s_iOP <= "000000";
189 s_iFunc <= "000000";
190
191 wait for gCLK_HPER*2;
192
193 -- Test case 17: Expected srl
194
195 s_iOP <= "000000";
196 s_iFunc <= "000010";
197
198 wait for gCLK_HPER*2;
199
200 -- Test case 18: Expected sra
201
202 s_iOP <= "000000";
203 s_iFunc <= "000011";

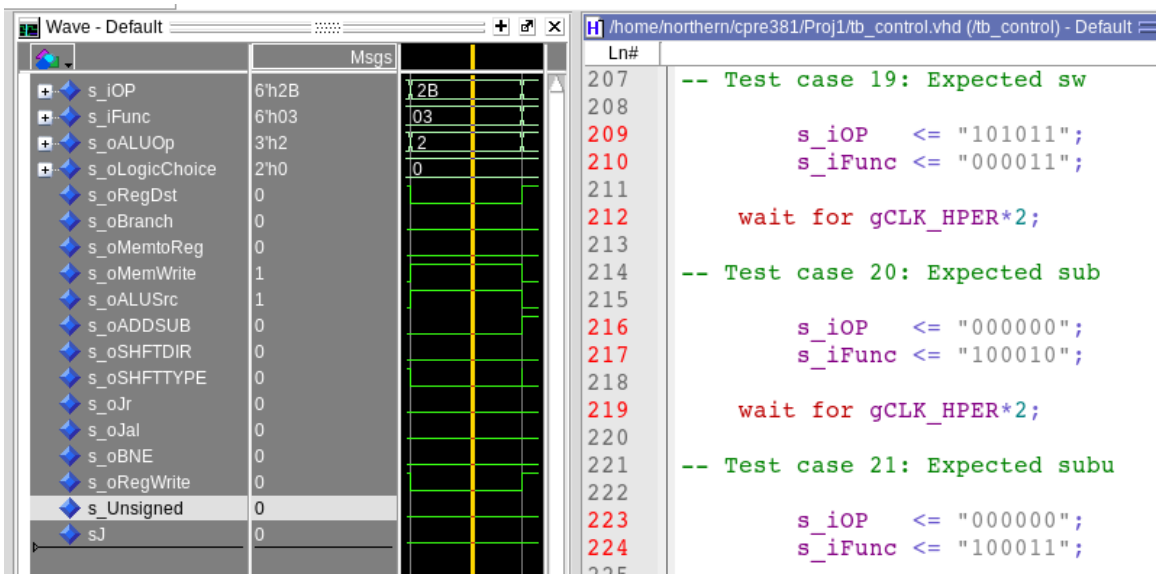
srl:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h02
s_oALUOp 3'h0
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
192
193 -- Test case 17: Expected srl
194
195 s_iOP <= "000000";
196 s_iFunc <= "000010";
197
198 wait for gCLK_HPER*2;
199
200 -- Test case 18: Expected sra
201
202 s_iOP <= "000000";
203 s_iFunc <= "000011";
204
205 wait for gCLK_HPER*2;
206
207 -- Test case 19: Expected sw
208
209 s_iOP <= "101011";
210 s_iFunc <= "000011";

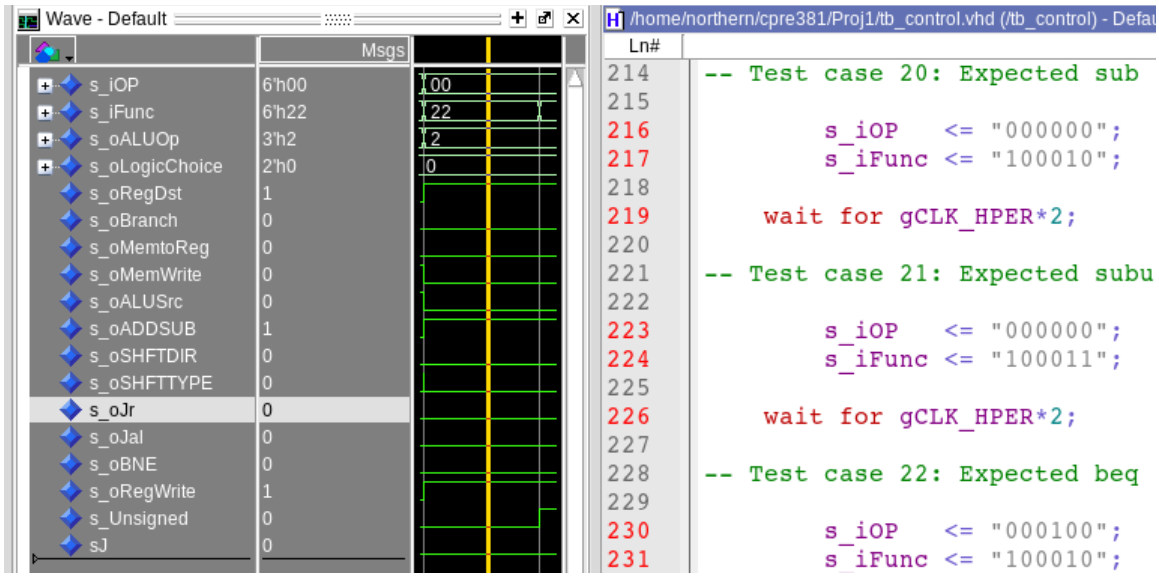
sra:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h03
s_oALUOp 3'h0
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 1
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
199
200 -- Test case 18: Expected sra
201
202 s_iOP <= "000000";
203 s_iFunc <= "000011";
204
205 wait for gCLK_HPER*2;
206
207 -- Test case 19: Expected sw
208
209 s_iOP <= "101011";
210 s_iFunc <= "000011";
211
212 wait for gCLK_HPER*2;
213
214 -- Test case 20: Expected sub
215
216 s_iOP <= "000000";

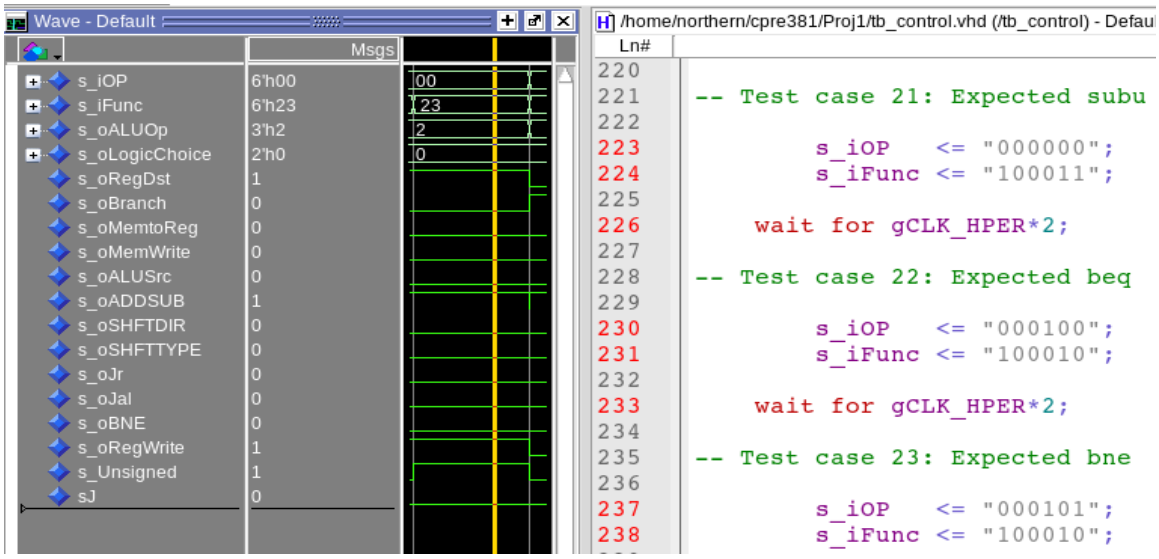
sw:


Wave - Default
Msgs
s_iOP 6'h2B
s_iFunc 6'h03
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 1
s_oALUSrc 1
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 0
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Default
Ln#
207 -- Test case 19: Expected sw
208
209 s_iOP <= "101011";
210 s_iFunc <= "000011";
211
212 wait for gCLK_HPER*2;
213
214 -- Test case 20: Expected sub
215
216 s_iOP <= "000000";
217 s_iFunc <= "100010";
218
219 wait for gCLK_HPER*2;
220
221 -- Test case 21: Expected subu
222
223 s_iOP <= "000000";
224 s_iFunc <= "100011";

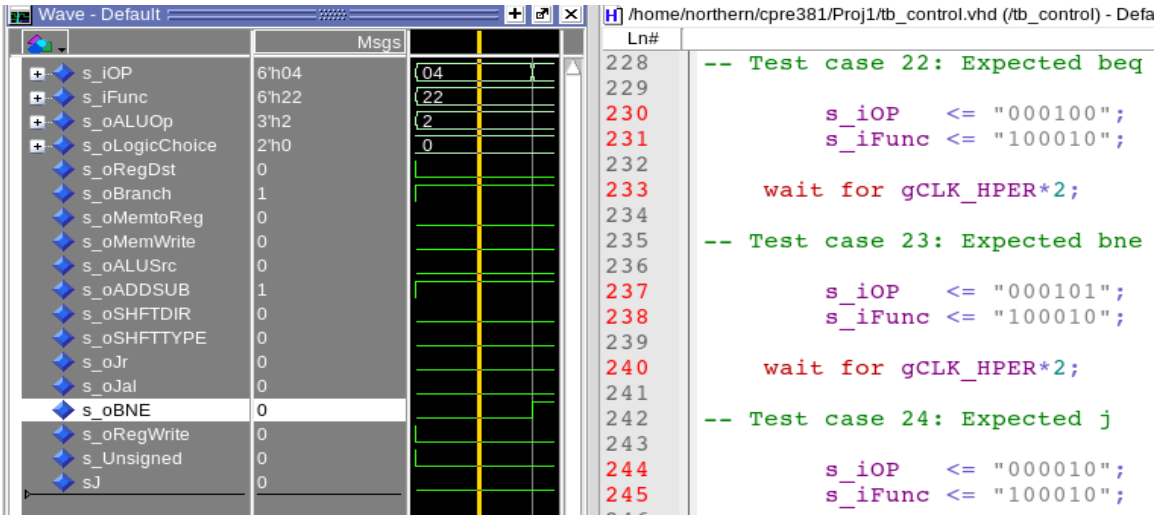
sub:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h22
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 1
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defau
Ln#
214 -- Test case 20: Expected sub
215
216 s_iOP <= "000000";
217 s_iFunc <= "100010";
218
219 wait for gCLK_HPER*2;
220
221 -- Test case 21: Expected subu
222
223 s_iOP <= "000000";
224 s_iFunc <= "100011";
225
226 wait for gCLK_HPER*2;
227
228 -- Test case 22: Expected beq
229
230 s_iOP <= "000100";
231 s_iFunc <= "100010";

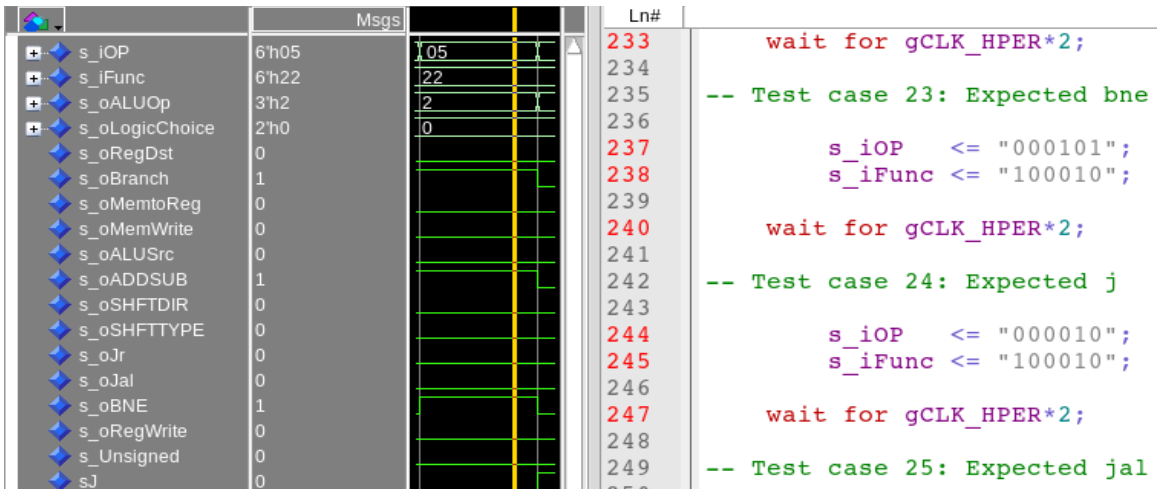
subu:


Wave - Default
Msgs
s_iOP 6'h00
s_iFunc 6'h23
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 1
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 1
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 1
s_Unsigned 1
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defau
Ln#
220
221 -- Test case 21: Expected subu
222
223 s_iOP <= "000000";
224 s_iFunc <= "100011";
225
226 wait for gCLK_HPER*2;
227
228 -- Test case 22: Expected beq
229
230 s_iOP <= "000100";
231 s_iFunc <= "100010";
232
233 wait for gCLK_HPER*2;
234
235 -- Test case 23: Expected bne
236
237 s_iOP <= "000101";
238 s_iFunc <= "100010";

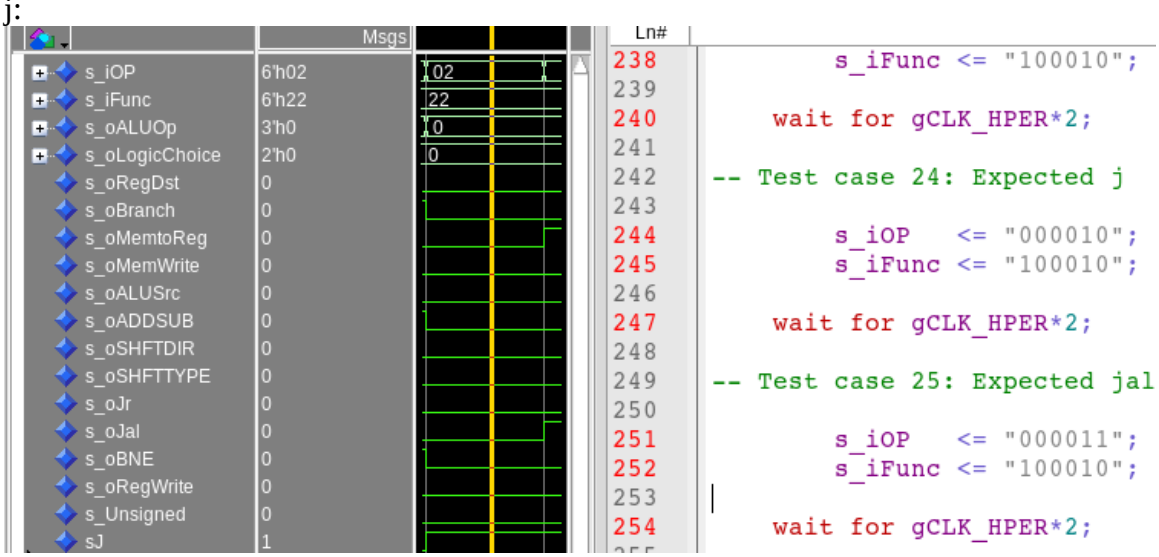
beq:


Wave - Default
Msgs
s_iOP 6'h04
s_iFunc 6'h22
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 1
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 1
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 0
s_Unsigned 0
sJ 0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Defa
Ln#
228 -- Test case 22: Expected beq
229
230 s_iOP <= "000100";
231 s_iFunc <= "100010";
232
233 wait for gCLK_HPER*2;
234
235 -- Test case 23: Expected bne
236
237 s_iOP <= "000101";
238 s_iFunc <= "100010";
239
240 wait for gCLK_HPER*2;
241
242 -- Test case 24: Expected j
243
244 s_iOP <= "000010";
245 s_iFunc <= "100010";

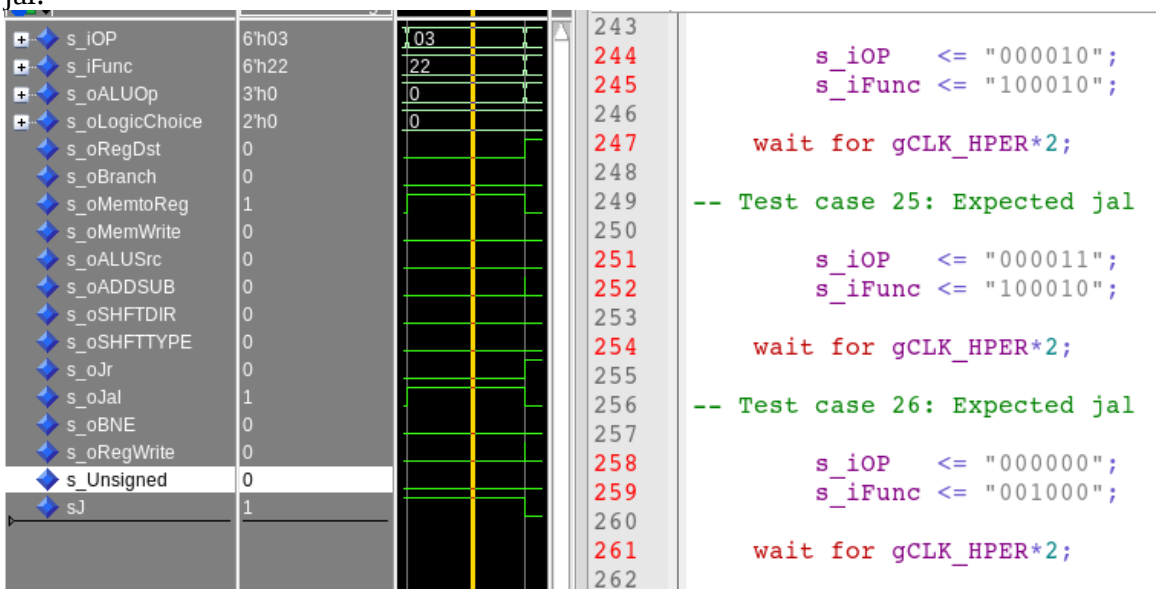
bne:


Msgs
s_iOP 6'h05
s_iFunc 6'h22
s_oALUOp 3'h2
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 1
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 1
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 1
s_oRegWrite 0
s_Unsigned 0
sJ 0
Ln#
233 wait for gCLK_HPER*2;
234
235 -- Test case 23: Expected bne
236
237 s_iOP <= "000101";
238 s_iFunc <= "100010";
239
240 wait for gCLK_HPER*2;
241
242 -- Test case 24: Expected j
243
244 s_iOP <= "000010";
245 s_iFunc <= "100010";
246
247 wait for gCLK_HPER*2;
248
249 -- Test case 25: Expected jal

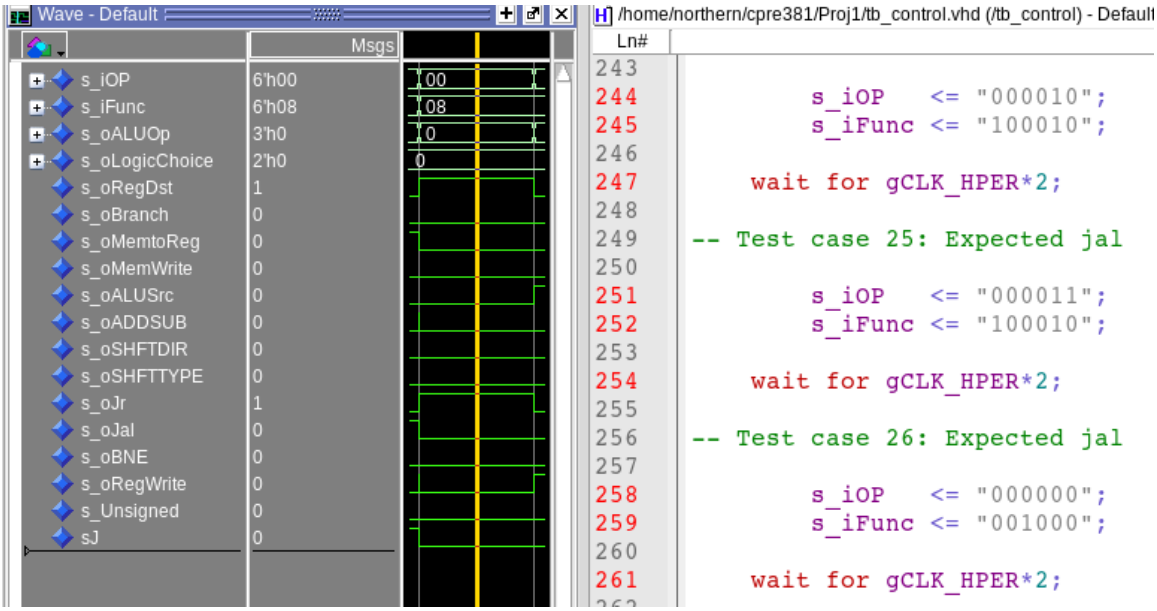
j:


Msgs
s_iOP 6'h02
s_iFunc 6'h22
s_oALUOp 3'h0
s_oLogicChoice 2'h0
s_oRegDst 0
s_oBranch 0
s_oMemtoReg 0
s_oMemWrite 0
s_oALUSrc 0
s_oADDSUB 0
s_oSHFTDIR 0
s_oSHFTTYPE 0
s_oJr 0
s_oJal 0
s_oBNE 0
s_oRegWrite 0
s_Unsigned 0
sJ 1
Ln#
238 s_iFunc <= "100010";
239
240 wait for gCLK_HPER*2;
241
242 -- Test case 24: Expected j
243
244 s_iOP <= "000010";
245 s_iFunc <= "100010";
246
247 wait for gCLK_HPER*2;
248
249 -- Test case 25: Expected jal
250
251 s_iOP <= "000011";
252 s_iFunc <= "100010";
253
254 wait for gCLK_HPER*2;

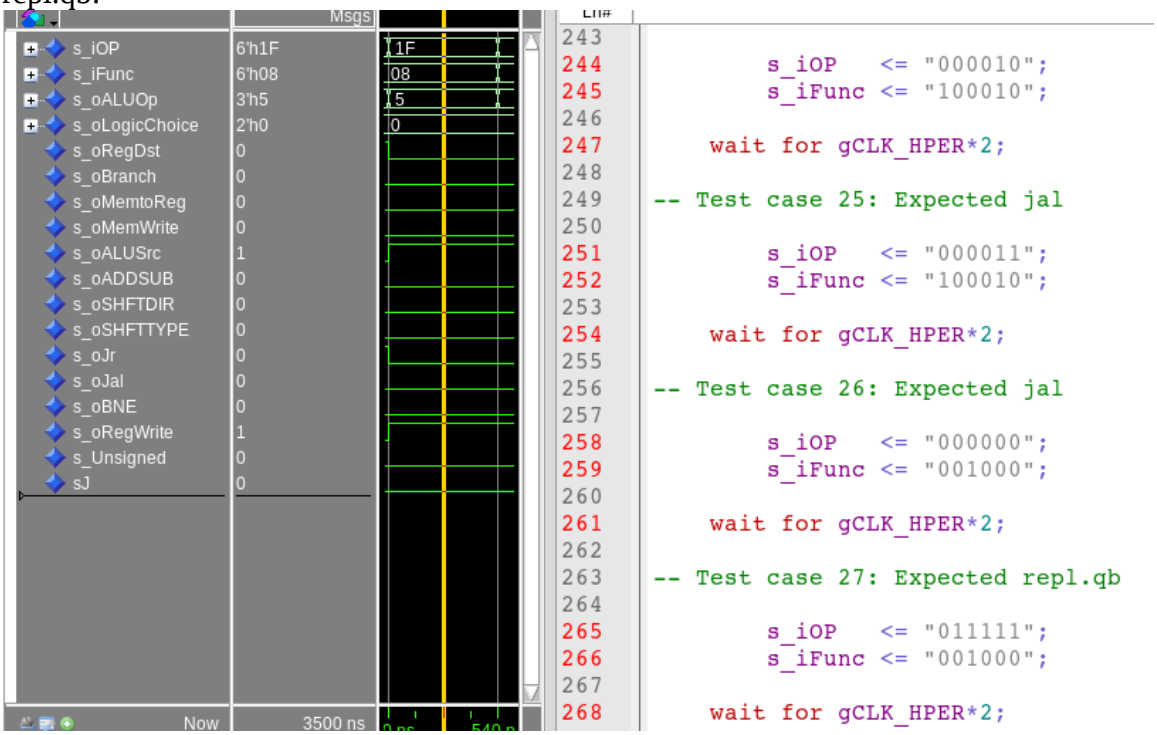
jal:



jr: --mistyped saying 26 was jal again but it was jr


Wave - Default
Msgs
s_iOP
s_iFunc
s_oALUOp
s_oLogicChoice
s_oRegDst
s_oBranch
s_oMemtoReg
s_oMemWrite
s_oALUSrc
s_oADDSUB
s_oSHFTDIR
s_oSHFTTYPE
s_oJr
s_oJal
s_oBNE
s_oRegWrite
s_Unsigned
sJ
6'h00
6'h08
3'h0
2'h0
00
08
0
/home/northern/cpre381/Proj1/tb_control.vhd (/tb_control) - Default
Ln#
243
244 s_iOP <= "000010";
245 s_iFunc <= "100010";
246
247 wait for gCLK_HPER*2;
248
249 -- Test case 25: Expected jal
250
251 s_iOP <= "000011";
252 s_iFunc <= "100010";
253
254 wait for gCLK_HPER*2;
255
256 -- Test case 26: Expected jal
257
258 s_iOP <= "000000";
259 s_iFunc <= "001000";
260
261 wait for gCLK_HPER*2;

repl.qb:


Msgs
s_iOP
s_iFunc
s_oALUOp
s_oLogicChoice
s_oRegDst
s_oBranch
s_oMemtoReg
s_oMemWrite
s_oALUSrc
s_oADDSUB
s_oSHFTDIR
s_oSHFTTYPE
s_oJr
s_oJal
s_oBNE
s_oRegWrite
s_Unsigned
sJ
6'h1F
6'h08
3'h5
2'h0
1F
08
5
0
Now
3500 ns
243
244 s_iOP <= "000010";
245 s_iFunc <= "100010";
246
247 wait for gCLK_HPER*2;
248
249 -- Test case 25: Expected jal
250
251 s_iOP <= "000011";
252 s_iFunc <= "100010";
253
254 wait for gCLK_HPER*2;
255
256 -- Test case 26: Expected jal
257
258 s_iOP <= "000000";
259 s_iFunc <= "001000";
260
261 wait for gCLK_HPER*2;
262
263 -- Test case 27: Expected repl.qb
264
265 s_iOP <= "011111";
266 s_iFunc <= "001000";
267
268 wait for gCLK_HPER*2;

[Part 2 (b.i)] What are the control flow possibilities that your instruction fetch logic must support? Describe these possibilities as a function of the different control flow-related instructions you are required to implement.

BNE: must take a Branch not equal input to an and gate andded with not Zero, this must activate the branch address or keep the normally incremented PCAddress.

BEQ: must take a Branch is Equal input to an and gate with ZERO, to then choose for a multiplexer whether to branch or stay on same PC Address.

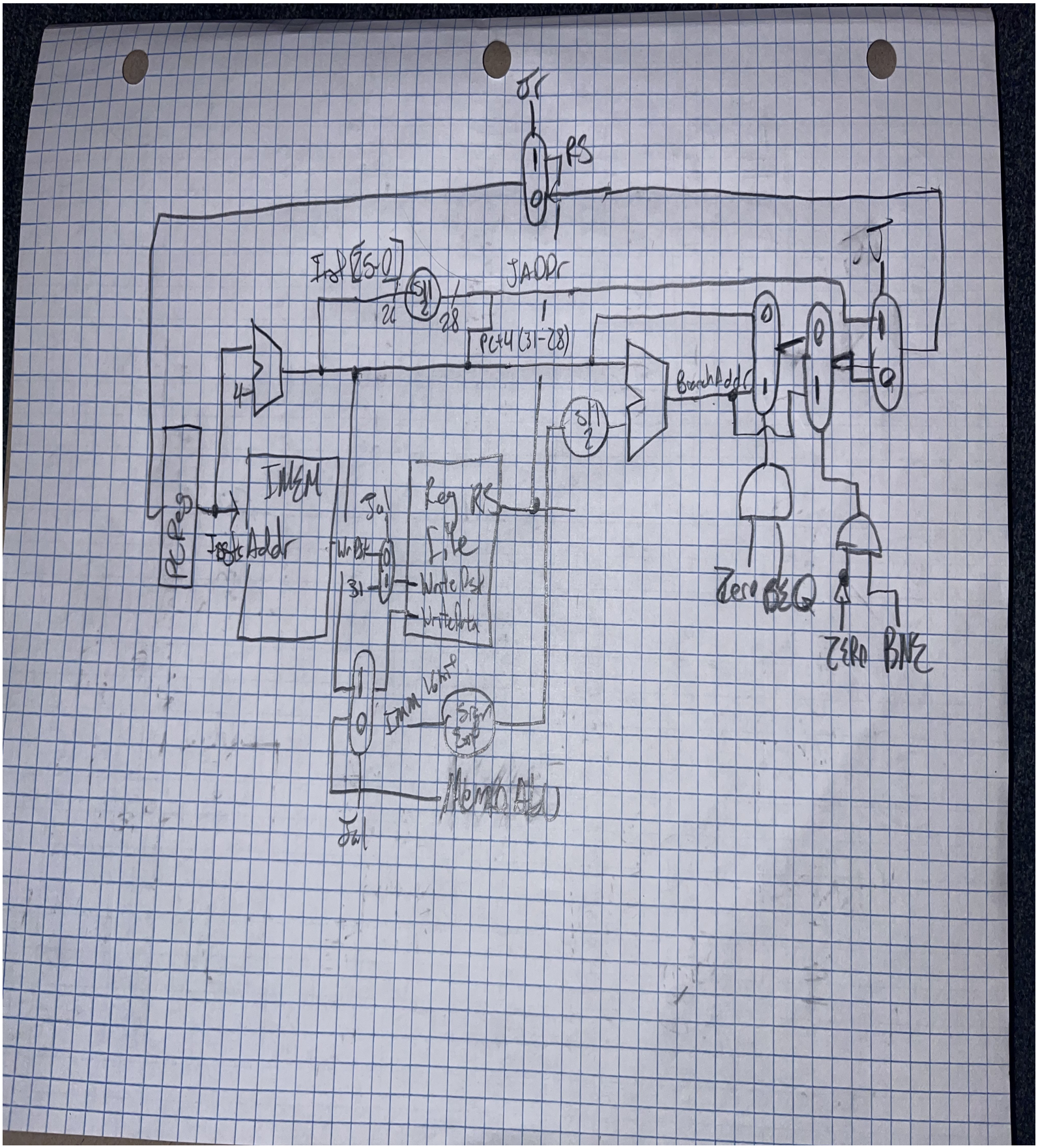
Jal: Jump and link must set register 31 to the PC Address incremented 4, by using a multiplexer on the write address port of the register file and porting the PC Address + 4 into a multiplexer to choose to take the output from the mem or ALU or the PCAddress and activates the PC Address when Jal is active to 1.

J: Jump runs to a multiplexer to choose between the given jump address extended by 2 0s and taking the first bits of the PC Address, or to keep the normal PC Address.

Jr: Jump Register jumps to the saved register address in RS usually register 31, Jr signal is used to choose between the Jr Address input, or the normal PC + 4 Address.

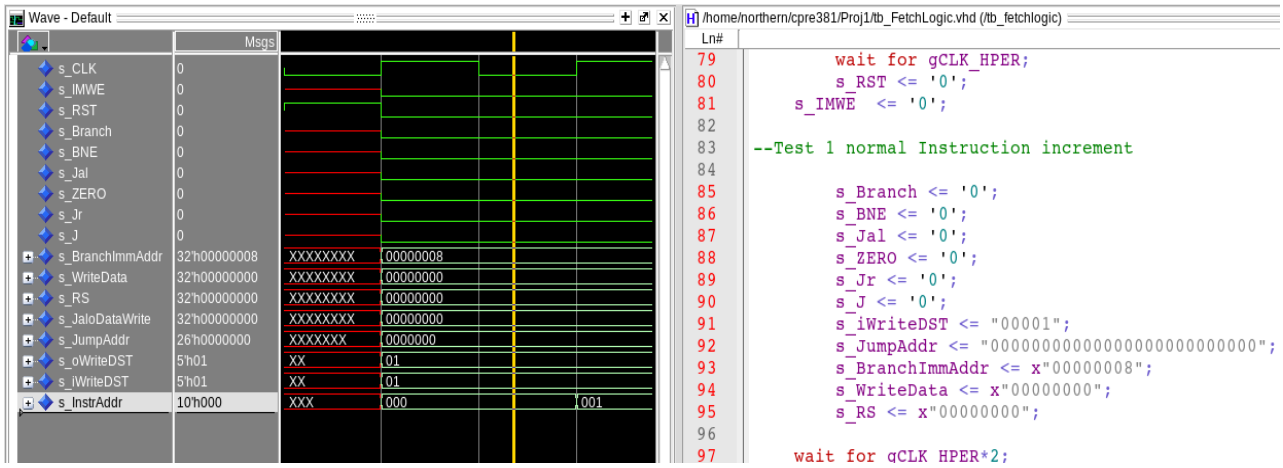
[Part 2 (b.ii)] Draw a schematic for the instruction fetch logic and any other datapath modifications needed for control flow instructions. What additional control signals are needed?

We needed BNE, BEQ, Jal, J, Jr to work out our control flow operations.

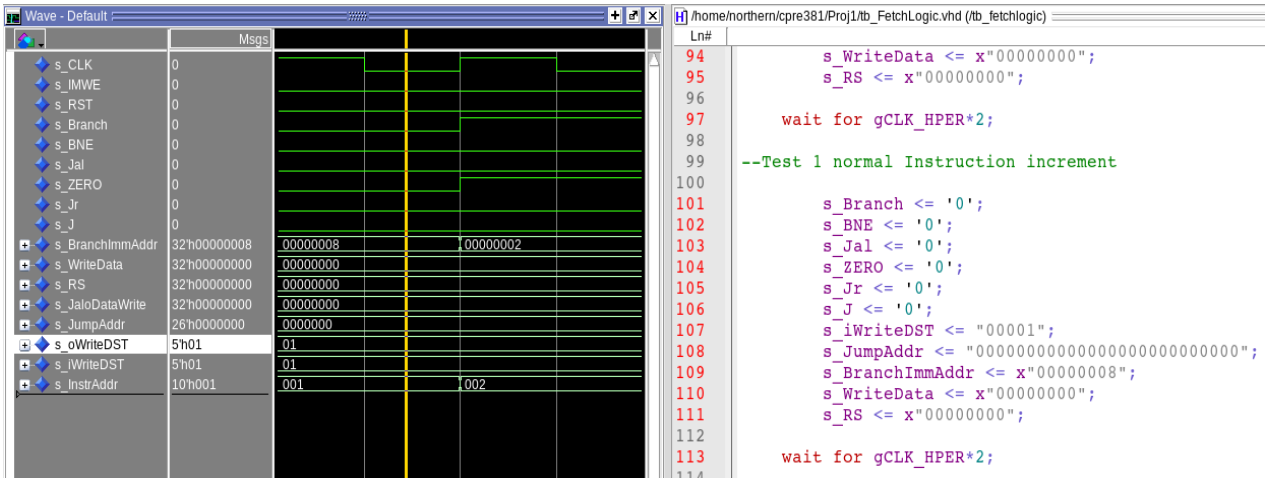

Jr
RS
Inst[25-0]
26
28
JAddr
PC+4[31-28]
4
IMEM
PC
InstrAddr
RegWr
Reg File
WriteRst
WriteData
RS
Ja
31
Imm16
Sign Ext
MemtoReg
Jal
BranchAddr
Zero
BEQ
ZERO
BNE

[Part 2 (b.iii)] Implement your new instruction fetch logic using VHDL. Use Modelsim to test your design thoroughly to make sure it is working as expected. Describe how the execution of the control flow possibilities corresponds to the Modelsim waveforms in your writeup.

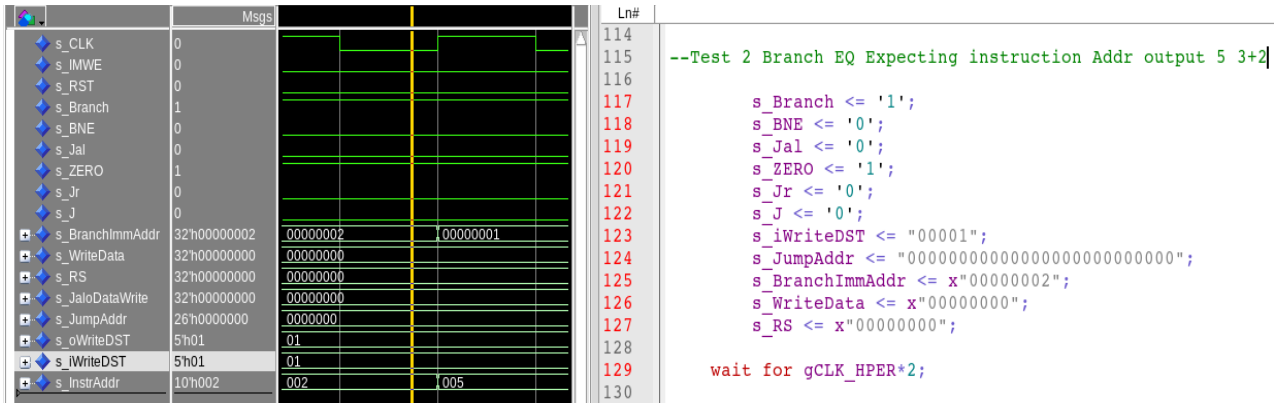
Normal Increment:



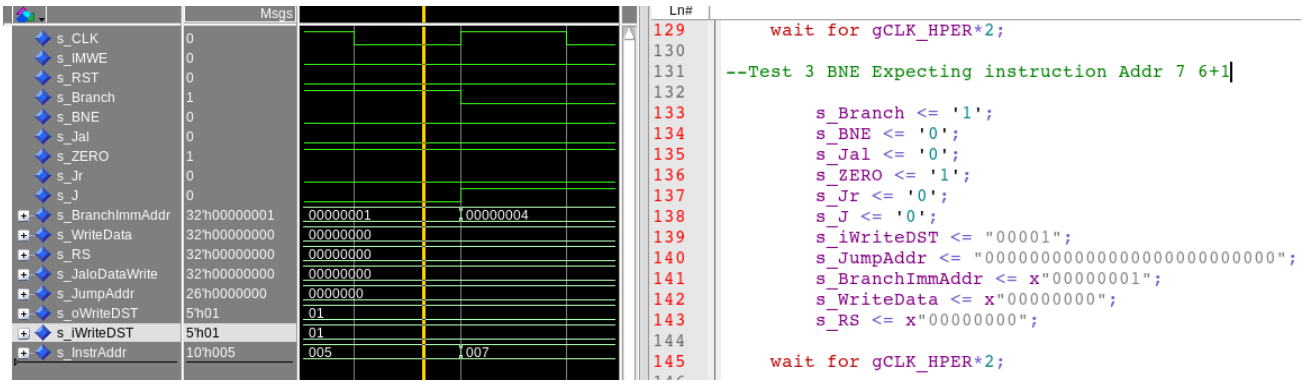
Normal Increment:



BEQ:


Msgs
s_CLK 0
s_IMWE 0
s_RST 0
s_Branch 1
s_BNE 0
s_Jal 0
s_ZERO 1
s_Jr 0
s_J 0
s_BranchImmAddr 32'h00000002 00000002 00000001
s_WriteData 32'h00000000 00000000
s_RS 32'h00000000 00000000
s_JaloDataWrite 32'h00000000 00000000
s_JumpAddr 26'h0000000 0000000
s_oWriteDST 5'h01 01
s_iWriteDST 5'h01 01
s_InstrAddr 10'h002 002 005
Ln#
114
115 --Test 2 Branch EQ Expecting instruction Addr output 5 3+2
116
117 s_Branch <= '1';
118 s_BNE <= '0';
119 s_Jal <= '0';
120 s_ZERO <= '1';
121 s_Jr <= '0';
122 s_J <= '0';
123 s_iWriteDST <= "00001";
124 s_JumpAddr <= "00000000000000000000000000";
125 s_BranchImmAddr <= x"00000002";
126 s_WriteData <= x"00000000";
127 s_RS <= x"00000000";
128
129 wait for gCLK_HPER*2;
130

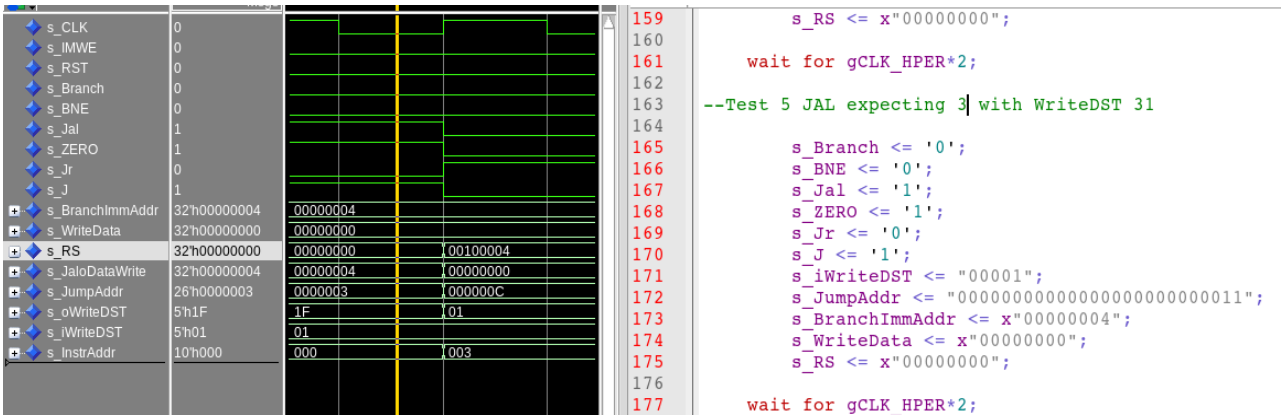
BNE:


Msgs
s_CLK 0
s_IMWE 0
s_RST 0
s_Branch 1
s_BNE 0
s_Jal 0
s_ZERO 1
s_Jr 0
s_J 0
s_BranchImmAddr 32'h00000001 00000001 00000004
s_WriteData 32'h00000000 00000000
s_RS 32'h00000000 00000000
s_JaloDataWrite 32'h00000000 00000000
s_JumpAddr 26'h0000000 0000000
s_oWriteDST 5'h01 01
s_iWriteDST 5'h01 01
s_InstrAddr 10'h005 005 007
Ln#
129 wait for gCLK_HPER*2;
130
131 --Test 3 BNE Expecting instruction Addr 7 6+1
132
133 s_Branch <= '1';
134 s_BNE <= '0';
135 s_Jal <= '0';
136 s_ZERO <= '1';
137 s_Jr <= '0';
138 s_J <= '0';
139 s_iWriteDST <= "00001";
140 s_JumpAddr <= "00000000000000000000000000";
141 s_BranchImmAddr <= x"00000001";
142 s_WriteData <= x"00000000";
143 s_RS <= x"00000000";
144
145 wait for gCLK_HPER*2;

J:

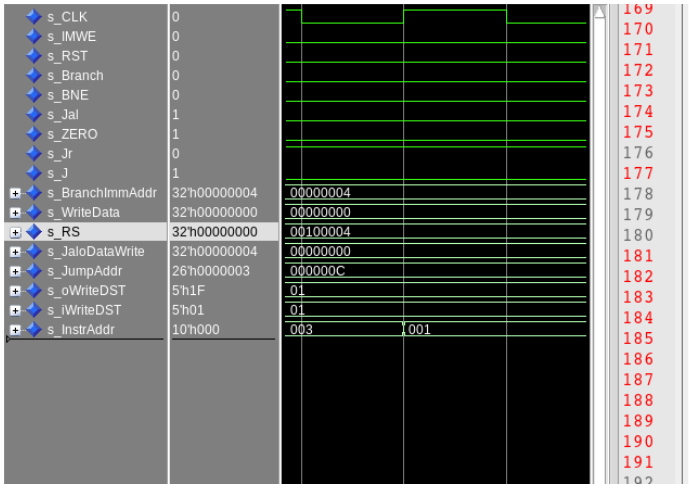

Msgs
s_CLK 0
s_IMWE 0
s_RST 0
s_Branch 0
s_BNE 0
s_Jal 0
s_ZERO 1
s_Jr 0
s_J 1
s_BranchImmAddr 32'h00000004 00000004
s_WriteData 32'h00000000 00000000
s_RS 32'h00000000 00000000
s_JaloDataWrite 32'h00000000 00000000 00000004
s_JumpAddr 26'h0000000 0000000 0000003
s_oWriteDST 5'h01 01 1F
s_iWriteDST 5'h01 01
s_InstrAddr 10'h007 007 000
Ln#
144
145 wait for gCLK_HPER*2;
146
147 --Test 4 Jump Expecting Addr Output 0
148
149 s_Branch <= '0';
150 s_BNE <= '0';
151 s_Jal <= '0';
152 s_ZERO <= '1';
153 s_Jr <= '0';
154 s_J <= '1';
155 s_iWriteDST <= "00001";
156 s_JumpAddr <= "00000000000000000000000000";
157 s_BranchImmAddr <= x"00000004";
158 s_WriteData <= x"00000000";
159 s_RS <= x"00000000";
160
161 wait for gCLK_HPER*2;

Jal:




```
159         s_RS <= x"00000000";
160
161     wait for gCLK_HPER*2;
162
163 --Test 5 JAL expecting 3 with WriteDST 31
164
165         s_Branch <= '0';
166         s_BNE <= '0';
167         s_Jal <= '1';
168         s_ZERO <= '1';
169         s_Jr <= '0';
170         s_J <= '1';
171         s_iWriteDST <= "00001";
172         s_JumpAddr <= "00000000000000000000000011";
173         s_BranchImmAddr <= x"00000004";
174         s_WriteData <= x"00000000";
175         s_RS <= x"00000000";
176
177     wait for gCLK_HPER*2;
```

Jr:




```
169         s_Jr <= '0';
170         s_J <= '1';
171         s_iWriteDST <= "00001";
172         s_JumpAddr <= "00000000000000000000000011";
173         s_BranchImmAddr <= x"00000004";
174         s_WriteData <= x"00000000";
175         s_RS <= x"00000000";
176
177     wait for gCLK_HPER*2;
178
179 --Test 6 JR expecting Address 1
180
181         s_Branch <= '0';
182         s_BNE <= '0';
183         s_Jal <= '0';
184         s_ZERO <= '0';
185         s_Jr <= '1';
186         s_J <= '0';
187         s_iWriteDST <= "00001";
188         s_JumpAddr <= "00000000000000000000001100";
189         s_BranchImmAddr <= x"00000004";
190         s_WriteData <= x"00000000";
191         s_RS <= x"00100004";
```

[Part 2 (c.i.1)] Describe the difference between logical (`srl`) and arithmetic (`sra`) shifts. Why does MIPS not have a `sla` instruction?

Srl fills in zeros in the shift, while sra fills in the sign bit. MIPS does not have a sla instruction, as the sign bit is only the most significant bit. Sla would most likely fill the least significant bit, which would change the value of the data being shifted in an undesired way.

[Part 2 (c.i.2)] In your writeup, briefly describe how your VHDL code implements both the arithmetic and logical shifting operations.

By using a control bit, the shifter can switch between filling with zeros and filling with the sign bit using a multiplexor.

[Part 2 (c.i.3)] In your writeup, explain how the right barrel shifter above can be enhanced to also support left shifting operations.

To support left shifting operations, all the right shifter would need to do is reverse the order of bits of the input, then do a right shift, and then reverse the output bits again. This would essentially create a left shift.

[Part 2 (c.i.4)] Describe how the execution of the different shifting operations corresponds to the Modelsim waveforms in your writeup.

[Part 2 (c.ii.1)] In your writeup, briefly describe your design approach, including any resources you used to choose or implement the design. Include at least one design decision you had to make.

We had to decide to combine our logic outputs in our ALU into one Logic Unit in order to save space on our ALU operation select multiplexer.

[Part 2 (c.ii.2)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.
Shown in our ALU testbench

[Part 2 (c.iii)] Draw a simplified, high-level schematic for the 32-bit ALU. Consider the following questions: how is Overflow calculated? How is Zero calculated? How is `slt` implemented?

[Part 2 (c.v)] Describe how the execution of the different operations corresponds to the Modelsim waveforms in your writeup.

Shown in our test bench

[Part 2 (c.viii)] justify why your test plan is comprehensive. Include waveforms that demonstrate your test programs functioning.

[Part 3] In your writeup, show the Modelsim output for each of the following tests, and provide a discussion of result correctness. It may be helpful to also annotate the waveforms directly.

[Part 3 (a)] Create a test application that makes use of every required arithmetic/logical instruction at least once. The application need not perform any particularly useful task, but it should demonstrate the full functionality of the processor (e.g., sequences of many instructions executed sequentially, 1 per cycle while data written into registers can be effectively retrieved and used by later instructions). Name this file Proj1_base_test.s.

[Part 3 (b)] Create and test an application which uses each of the required control-flow instructions and has a call depth of at least 5 (i.e., the number of activation records on the stack is at least 4). Name this file Proj1_cf_test.s.

[Part 3 (c)] Create and test an application that sorts an array with *N* elements using the BubbleSort algorithm (link). Name this file Proj1_bubblesort.s.

[Part 4] report the maximum frequency your processor can run at and determine what your critical path is. Draw this critical path on top of your top-level schematics. What components would you focus on to improve the frequency?