ESE345 Project Report

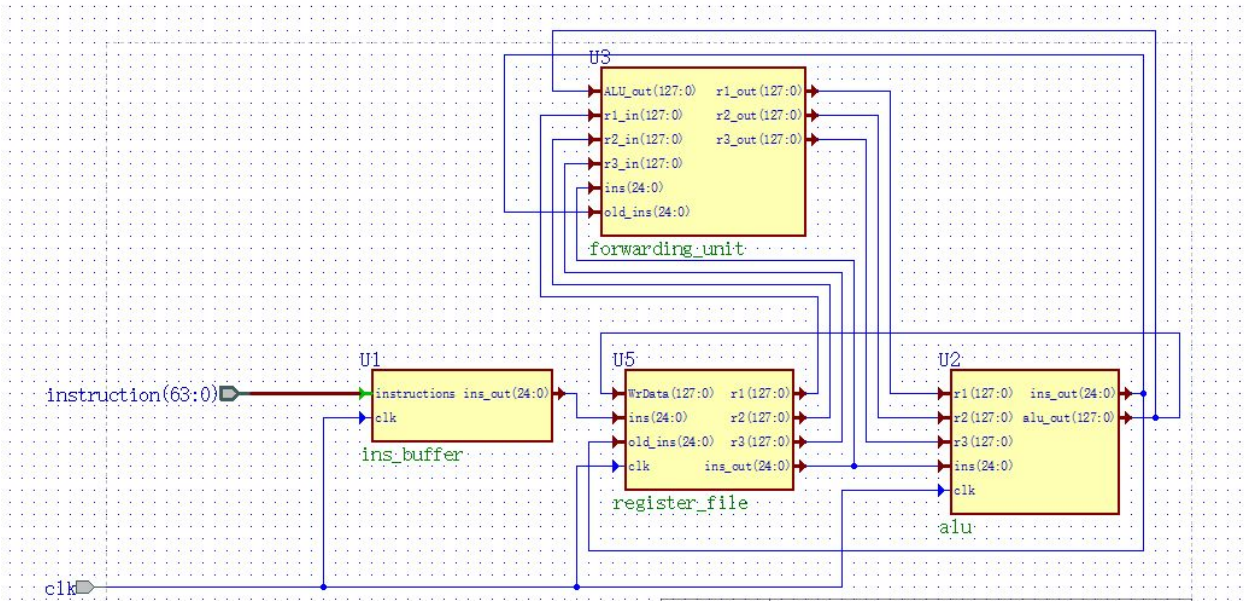
Fengwei Zhang #111252554

Gavin Zhou #111586947

1. Project Goal

This objective of the project is to learn the use of VHDL/Verilog hardware description language and modern CAD tools for the hierarchical gate-level and dataflow/RTL design of the 4-stage pipelined multimedia unit with a reduced set of multimedia instructions similar to those in the Sony Cell SPU and Intel SSE architectures.

2. Multimedia unit block diagram



3. Design procedure

3.1. ALU

In this module, there are five input which are 128 bit vector r1, r2, r3 and 25 bit instruction and clk, and two outputs 25 bit ins and calculated 128 bit alu_out. Case statement based on instruction's 25 to 15 bit is used to tell which instruction is in this

cycle. The reason why there is a ins_out is like transferring the control signal to register file to tell register which register have to write into.

For the load instruction, first give r1's value to the alu_out and then load immediate to the alu_out.

For the R4 type instruction, temp1 and temp2 which are short 32 bit vector variable and temp3 and temp4 which are long 64 bit vector variable are used to get the temporary product of rs2 and rs3 and sum of product of rs2 and rs3 and rs1. There might be saturation when add and subtract. For the add saturation case, there are two cases, one is positive number plus positive number to get a positive saturation and negative number subtract negative numbers to get a negative saturation. For the subtract saturation case, there are still two cases which are positive number subtract negative numbers to get a positive saturation and negative number subtract positive number to get a negative saturation.

For the R3 type instruction, there are some or, xor, rotate, left shift and right shift instruction which can use pre-defined function or, xor, ror, sll and srl. Saturation cases are like the R4 type instruction, while in the MSGN instruction there is only one saturation case which is the for the same number of bit maximum negative number is bigger than the maximum positive number.

3.2. Instruction Buffer

Declare a 64 capacity array to store 64 instructions. Every PC count 1 more then fetch next instruction.

3.3. Register File

There are write enable control signal to determine should write into register. The old_ins is transferred from alu to determine in last cylce which register have to write into.

3.4. Forwarding Unit

All instructions takes 4 clock cycles to complete, which will cause data hazard if an instruction uses a register that is recently modified in the earlier 2 clock cycles. The forwarding unit will detect this situation and delivers the most recent data of that register to the input of ALU to avoid data hazard.

3.5. Pipeline Unit

Pipeline unit is a structural unit in the project. It works as a bridge between each unit. Signals or Variables will be declared to store the data and instructions output

from one unit and send them to the next unit's inputs.These signals can be seen as the pipeline registers, they have the same function.

4. Testbenches

Imported two new libraries STD.textio.all and ieee.std_logic_textio.all to use built-in function to read txt file generated by assembly to binary instruction converter program written in Java. And still write the alu_out 128 bit data into txt file.

5. Conclusions

The final design was tested and verified for various input data and different op-codes using different inputs in the testbench. All the instructions functions as expected and the results generated match the theoretical results. So it was determined that the final design is a success.

Test Instruction List:

li r1,0,0000000000000001

li r1,1,0000000000000010

li r1,2,0000000000000011

li r1,3,0000000000000100

li r1,4,0000000000000101

li r1,5,0000000000000110

li r1,6,0000000000000111

li r1,7,0000000000001000

li r2,0,0000000000000001

li r2,1,0000000000000010

li r2,2,0000000000000011

li r2,3,0000000000000100

li r2,4,0000000000000101

li r2,5,0000000000000110

li r2,6,0000000000000111

li r2,7,0000000000001000

li r3,0,0000000000000001

li r3,1,0000000000000010

li r3,2,0000000000000011

li r3,3,0000000000000100

li r3,4,0000000000000101

li r3,5,0000000000000110

li r3,6,0000000000000111

li r3,7,0000000000001000

IAL r4,r1,r2,r3

IAH r5,r1,r2,r3

ISL r6,r1,r2,r3

ISH r7,r1,r2,r3

LAL r8,r1,r2,r3

LAH r9,r1,r2,r3

LSL r10,r1,r2,r3

LSH r11,r1,r2,r3

A r12,r1,r2

AH r13,r1,r2

AHS r14,r1,r2

AND r15,r1,r2

BCW r16,r1,r2

CLZ r17,r1,r2

MAX r18,r1,r2

MIN r19,r1,r2

MSGN r20,r1,r2

MPYU r21,r1,r2

OR r22,r1,r2

POPCNTH r23,r1,r2

ROT r24,r1,r2

ROTW r25,r1,r2

SHLHI r26,r1,r2

SFH r27,r1,r2

SFW r28,r1,r2

SFHS r29,r1,r2

XOR r30,r1,r2

Opcode after transformed:

00000000000000000000100001

00010000000000000001000001

00100000000000000001100001

00110000000000000010000001

01000000000000000010100001

01010000000000000011000001

01100000000000000011100001

01110000000000000100000001

00000000000000000000100010

00010000000000000001000010

00100000000000000001100010

00110000000000000010000010

01000000000000000010100010

01010000000000000011000010

01100000000000000011100010

01110000000000000100000010

00000000000000000000100011

00010000000000000001000011

00100000000000000001100011

00110000000000000010000011

01000000000000000010100011

01010000000000000011000011

01100000000000000011100011

01110000000000000100000011

10000000110001000001001000

10001000110001000001001010

10010000110001000001001100

10011000110001000001001110

10100000110001000001010000

10101000110001000001010010

10110000110001000001010100

10111000110001000001010110

11000000010001000001011000

11000000100001000001011010

11000000110001000001011100

11000000111000100000111110

11000001010001000001100000

11000001100001000001100010

11000001110001000001100100

11000010000001000001100110

11000010010001000001101000

11000010100001000001101010

11000010110001000001101100

11000011000001000001101110

11000011010001000001110000

11000011100001000001110010

11000011110001000001110100

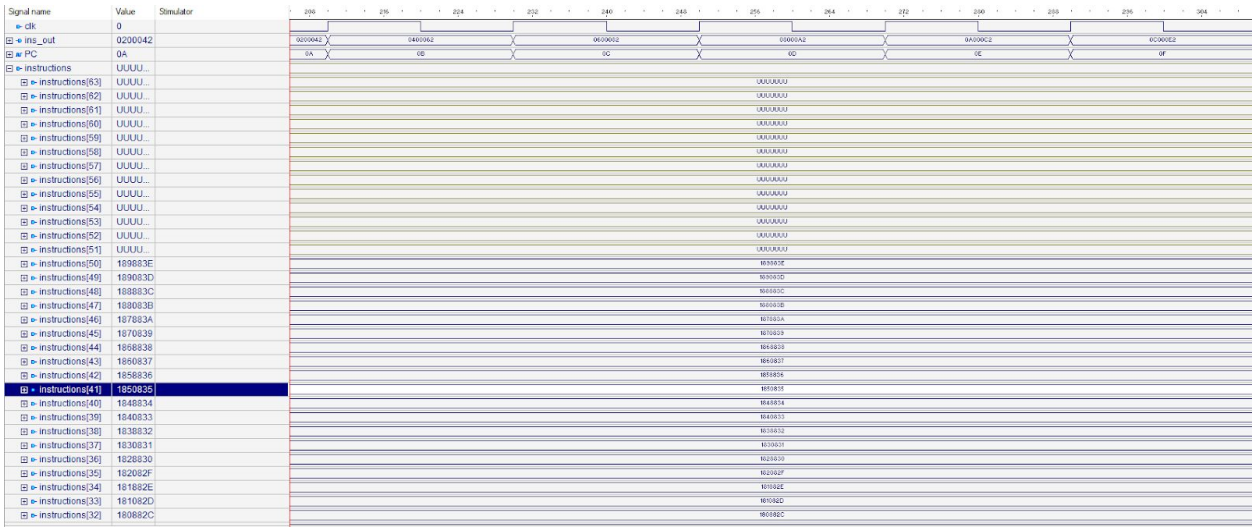
11000100000001000000111011

11000100010001000000111100

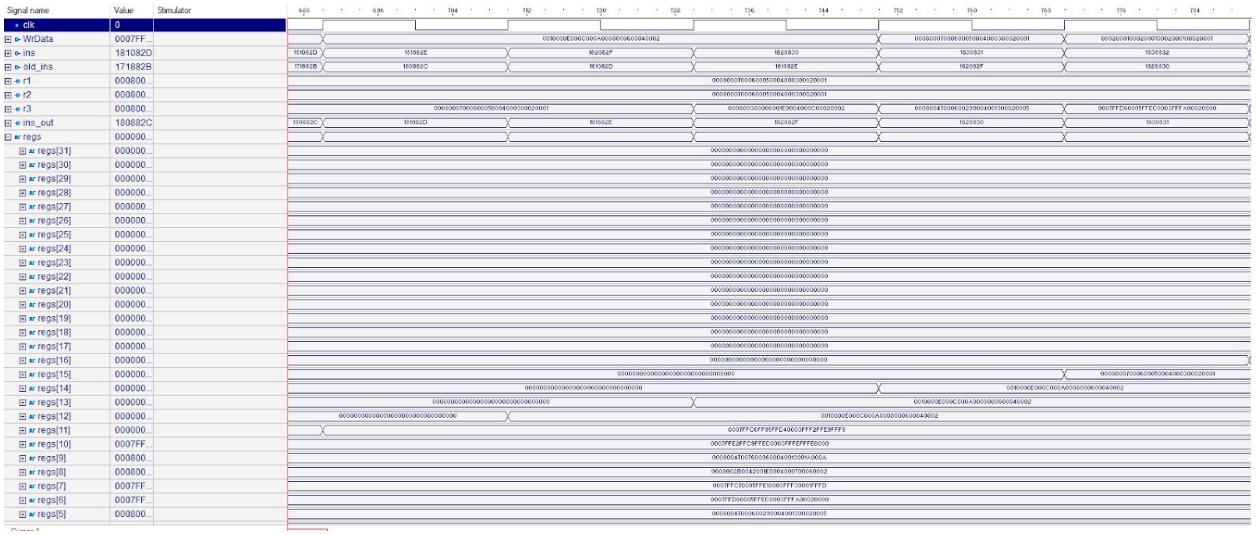
11000100100001000000111101

11000100110001000000111110

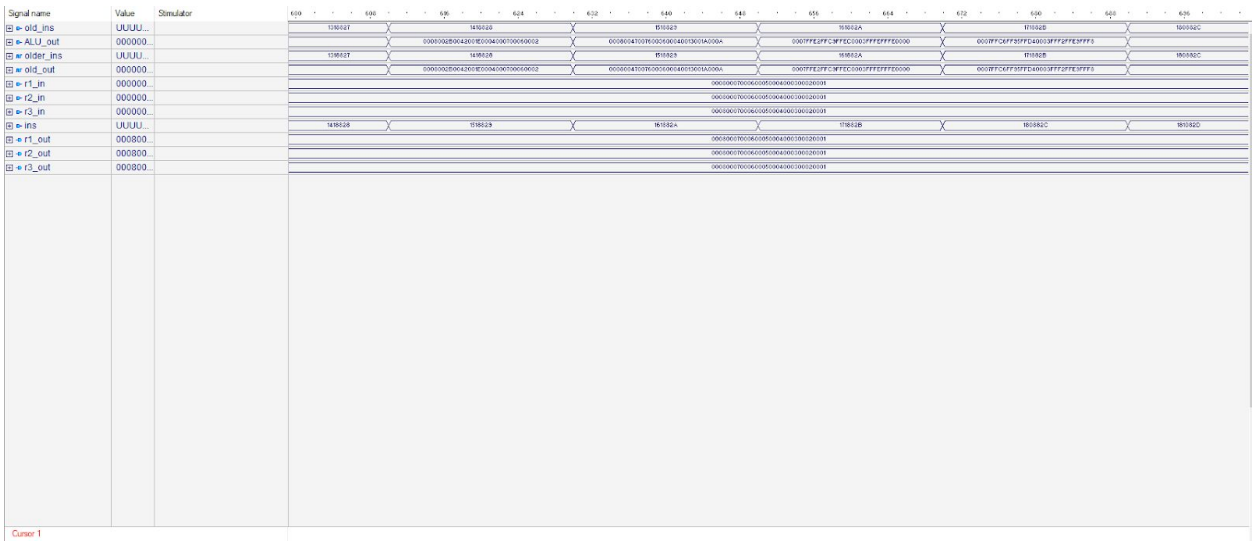
Waveform of Instruction Buffer during working(some instruction register is not used)



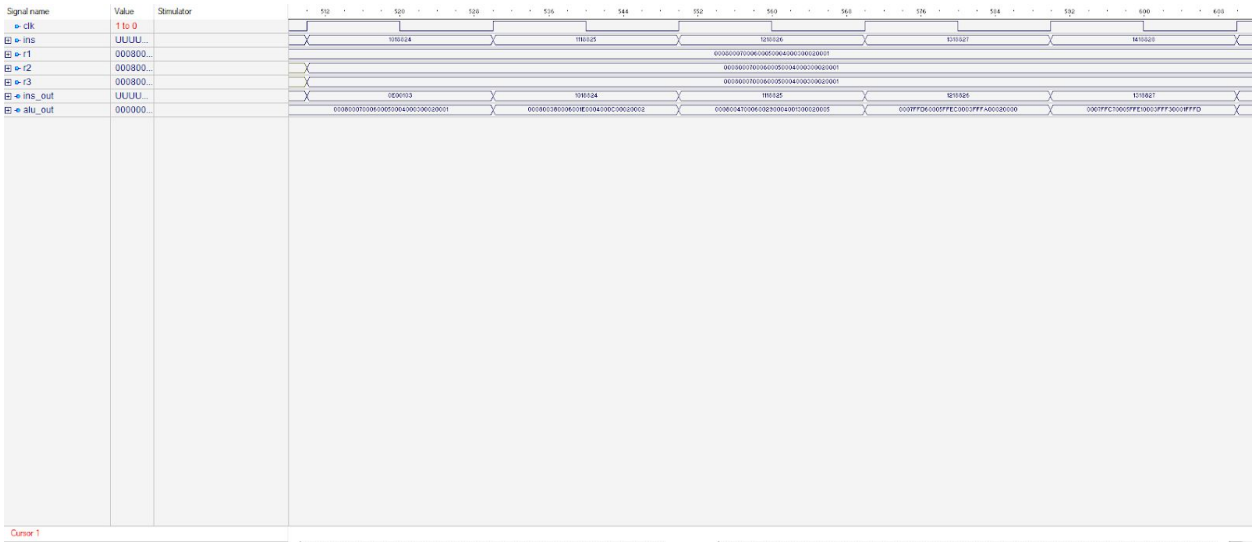
Waveform of Register File during working



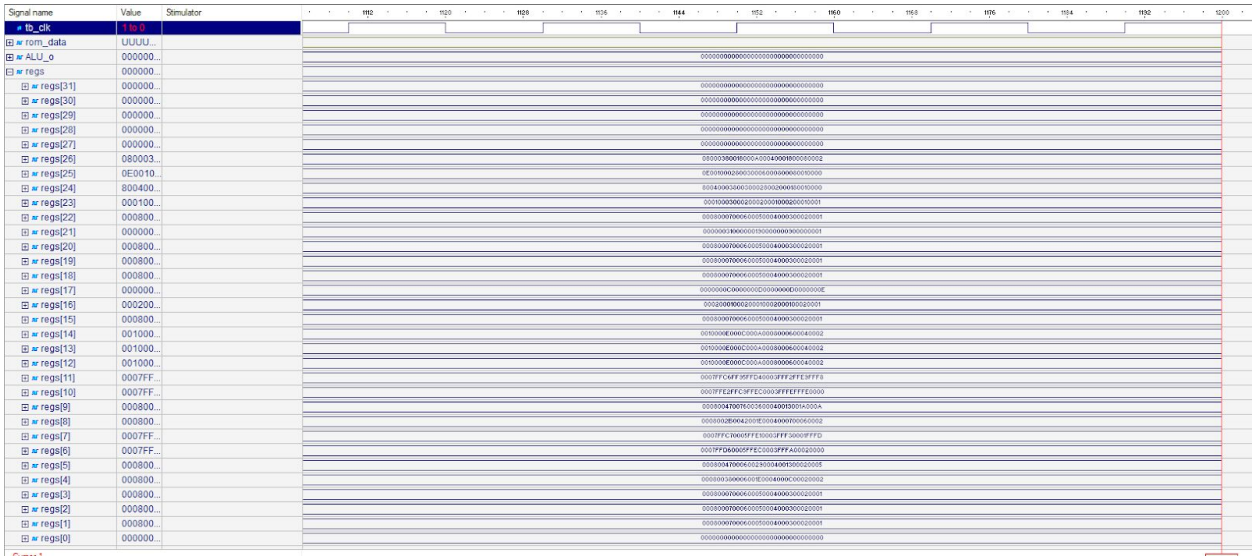
Waveform of forwarding unit during working



Waveform of ALU during working



Waveform of final results in the registers


Signal name
Value
Stimulator
tb_clk
rom_data
ALU_o
regs
regs[31]
regs[30]
regs[29]
regs[28]
regs[27]
regs[26]
regs[25]
regs[24]
regs[23]
regs[22]
regs[21]
regs[20]
regs[19]
regs[18]
regs[17]
regs[16]
regs[15]
regs[14]
regs[13]
regs[12]
regs[11]
regs[10]
regs[9]
regs[8]
regs[7]
regs[6]
regs[5]
regs[4]
regs[3]
regs[2]
regs[1]
regs[0]

Final result in register:

00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
08000380018000A00040001800080002
0E0010002800300060008000800100000
80040003800300028002000180010000
00010003000200020001000200010001
00080007000600050004000300020001
00000031000000130000000900000001
00080007000600050004000300020001
00080007000600050004000300020001
00080007000600050004000300020001
0000000C0000000D0000000D0000000E
00020001000200010002000100020001
00080007000600050004000300020001
0010000E000C000A00080006000400002
0010000E000C000A00080006000400002
0010000E000C000A00080006000400002
0007FFC6FF95FFD40003FFF2FFE3FFF8
0007FFE2FFC3FFEC0003FFFEFFFE0000
00080047007600360004001300 1A000A
0008002B0042001E00040007000600002
0007FFC70005FFE10003FFF30001FFFD
0007FFD60005FFEC0003FFFA00020000
00080047000600290004001300020005
00080038000600 1E0004000C00020002
00080007000600050004000300020001
00080007000600050004000300020001
00080007000600050004000300020001
00000000000000000000000000000000

Expected Results:

000800380006001E0004000C00020002

000800470006002900040013000200005

0007FFD60005FFEC0003FFFA00020000

0007FFC70005FFE10003FFF30001FFFD

0008002B0042001E0004000700060002

000800470076003600040013001A000A

0007FFE2FFC9FFEC0003FFFEFFFE0000

0007FFC6FF95FFD40003FFF2FFE9FFF8

0010000E000C000A0008000600040002

0010000E000C000A0008000600040002

0010000E000C000A0008000600040002

000800070006000500040003000200001

000200010002000100020001000200001

0000000C0000000D0000000D0000000E

000800070006000500040003000200001

000800070006000500040003000200001

000800070006000500040003000200001

000000310000001900000009000000001

000800070006000500040003000200001

000100030002000200010002000100001

800400038003000280020001800100000

0E00100028003000600080008001000 0

08000380800000A00040001800080002

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

00000000000000000000000000000000

File: f:/project345/src/alu.vhd (/file_io_tb/UUT/u3)

```
1
2     library ieee;
3     use ieee.std_logic_1164.all;
4     use ieee.numeric_std.all;
5     --use ieee.std_logic_arith.all;
6
7     entity alu is
8         port(
9         r1: in std_logic_vector(127 downto 0);
10        r2: in std_logic_vector(127 downto 0);
11        r3: in std_logic_vector(127 downto 0);
12        ins: in std_logic_vector(24 downto 0);
13        clk : in std_logic;
14        ins_out: out std_logic_vector(24 downto 0);
15        alu_out: out std_logic_vector(127 downto 0)
16        );
17    end alu;
18
19    architecture behavioral of alu is
20    begin
21        process(r1, r2, r3, ins,clk)
22            variable temp1, temp2: std_logic_vector(31 downto 0);
23            variable temp3, temp4: std_logic_vector(63 downto 0);
24            variable zero count : integer;
25            variable one_count : integer;
26            variable bits_of_shift : integer;
27        begin
28            if rising_edge(clk) then
29
30            ins_out<=ins;
31
32            ------------------4.1 load imm instruction-------------------------
33            if ins(24) = '0' then
34                if ins(23 downto 21) = "000" then
35                alu out <= r1;
36                alu out(15 downto 0) <= ins(20 downto 5);
37                elsif ins(23 downto 21) = "001" then
38                alu_out <= r1;
39                alu_out(31 downto 16) <= ins(20 downto 5);
40                elsif ins(23 downto 21) = "010" then
41                alu_out <= r1;
42                alu_out(47 downto 32) <= ins(20 downto 5);
43                elsif ins(23 downto 21) = "011" then
44                alu out <= r1;
45                alu out(63 downto 48) <= ins(20 downto 5);
46                elsif ins(23 downto 21) = "100" then
47                alu out <= r1;
48                alu out(79 downto 64) <= ins(20 downto 5);
49                elsif ins(23 downto 21) = "101" then
50                alu_out <= r1;
51                alu_out(95 downto 80) <= ins(20 downto 5);
52                elsif ins(23 downto 21) = "110" then
53                alu_out <= r1;
54                alu_out(111 downto 96) <= ins(20 downto 5);
55                elsif ins(23 downto 21) = "111" then
56                alu out <= r1;
57                alu out(127 downto 112) <= ins(20 downto 5);
58                end if;
59
60            ------------------4.2 R4 instruction--------------------------
61            -----------saturation??
62                elsif ins(24 downto 23) = "10" then
63
64                    case ins(22 downto 20) is
65                        -----Signed Integer Multiply-Add Low with
      Saturation------------------
```

```
66                          when "000" =>
67                          temp1:=std_logic_vector(signed(r2(111 downto 96))* signed(r3(111
    downto 96)));
68                          temp2:=std_logic_vector(signed(temp1) + signed(r1(127 downto 96)));
69                          if temp2(31)='1' and temp1(31)='0' and r1(127)='0' then
70                              alu_out(127 downto 96)<=(others=>'1');
71                              alu_out(127)<='0';
72                          elsif temp2(31)='0' and temp1(31)='1' and r1(127)='1' then
73                              alu_out(127 downto 96)<=(others=>'0');
74                              alu_out(127)<='1';
75                          else
76                              alu_out(127 downto 96)<=temp2;
77                          end if;
78
79                          temp1:=std_logic_vector(signed(r2(79 downto 64))* signed(r3(79
    downto 64)));
80                          temp2:=std_logic_vector(signed(temp1) + signed(r1(95 downto 64)));
81                          if temp2(31)='1' and temp1(31)='0' and r1(95)='0' then
82                              alu_out(95 downto 64)<=(others=>'1');
83                              alu_out(95)<='0';
84                          elsif temp2(31)='0' and temp1(31)='1' and r1(95)='1' then
85                              alu_out(95 downto 64)<=(others=>'0');
86                              alu_out(95)<='1';
87                          else
88                              alu_out(95 downto 64)<=temp2;
89                          end if;
90
91                          temp1:=std_logic_vector(signed(r2(47 downto 32))* signed(r3(47
    downto 32)));
92                          temp2:=std_logic_vector(signed(temp1) + signed(r1(63 downto 32)));
93                          if temp2(31)='1' and temp1(31)='0' and r1(63)='0' then
94                              alu_out(63 downto 32)<=(others=>'1');
95                              alu_out(63)<='0';
96                          elsif temp2(31)='0' and temp1(31)='1' and r1(63)='1' then
97                              alu_out(63 downto 32)<=(others=>'0');
98                              alu_out(63)<='1';
99                          else
100                             alu_out(63 downto 32)<=temp2;
101                         end if;
102
103                         temp1:=std_logic_vector(signed(r2(15 downto 0))* signed(r3(15 downto
    0)));
104                         temp2:=std_logic_vector(signed(temp1) + signed(r1(31 downto 0)));
105                         if temp2(31)='1' and temp1(31)='0' and r1(31)='0' then
106                             alu_out(31 downto 0)<=(others=>'1');
107                             alu_out(31)<='0';
108                         elsif temp2(31)='0' and temp1(31)='1' and r1(31)='1' then
109                             alu_out(31 downto 0)<=(others=>'0');
110                             alu_out(31)<='1';
111                         else
112                             alu_out(31 downto 0)<=temp2;
113                         end if;
114                         --------Signed Integer Multiply-Add High with
    Saturation----------------
115                         when "001" =>
116                         temp1:=std_logic_vector(signed(r2(127 downto 112))* signed(r3(127
    downto 112)));
117                         temp2:=std_logic_vector(signed(temp1) + signed(r1(127 downto 96)));
118                         if temp2(31)='1' and temp1(31)='0' and r1(127)='0' then
119                             alu_out(127 downto 96)<=(others=>'1');
120                             alu_out(127)<='0';
121                         elsif temp2(31)='0' and temp1(31)='1' and r1(127)='1' then
122                             alu_out(127 downto 96)<=(others=>'0');
123                             alu_out(127)<='1';
124                         else
125                             alu_out(127 downto 96)<=temp2;
```

```
126                         end if;
127
128                         temp1:=std_logic_vector(signed(r2(95 downto 80))* signed(r3(95
    downto 80)));
129                         temp2:=std_logic_vector(signed(temp1) + signed(r1(95 downto 64)));
130                         if temp2(31)='1' and temp1(31)='0' and r1(95)='0' then
131                             alu_out(95 downto 64)<=(others=>'1');
132                             alu_out(95)<='0';
133                         elsif temp2(31)='0' and temp1(31)='1' and r1(95)='1' then
134                             alu_out(95 downto 64)<=(others=>'0');
135                             alu_out(95)<='1';
136                         else
137                             alu_out(95 downto 64)<=temp2;
138                         end if;
139
140                         temp1:=std_logic_vector(signed(r2(63 downto 48))* signed(r3(63
    downto 48)));
141                         temp2:=std_logic_vector(signed(temp1) + signed(r1(63 downto 32)));
142                         if temp2(31)='1' and temp1(31)='0' and r1(63)='0' then
143                             alu_out(63 downto 32)<=(others=>'1');
144                             alu_out(63)<='0';
145                         elsif temp2(31)='0' and temp1(31)='1' and r1(63)='1' then
146                             alu_out(63 downto 32)<=(others=>'0');
147                             alu_out(63)<='1';
148                         else
149                             alu_out(63 downto 32)<=temp2;
150                         end if;
151
152                         temp1:=std_logic_vector(signed(r2(31 downto 16))* signed(r3(31
    downto 16)));
153                         temp2:=std_logic_vector(signed(temp1) + signed(r1(31 downto 0)));
154                         if temp2(31)='1' and temp1(31)='0' and r1(31)='0' then
155                             alu_out(31 downto 0)<=(others=>'1');
156                             alu_out(31)<='0';
157                         elsif temp2(31)='0' and temp1(31)='1' and r1(31)='1' then
158                             alu_out(31 downto 0)<=(others=>'0');
159                             alu_out(31)<='1';
160                         else
161                             alu_out(31 downto 0)<=temp2;
162                         end if;
163                         ----------Signed Integer Multiply-Subtract Low with
    Saturation-------------
164                         when "010" =>
165                         temp1:=std_logic_vector(signed(r2(111 downto 96))*signed(r3(111
    downto 96)));
166                         temp2:=std_logic_vector(signed(r1(127 downto 96))-signed(temp1));
167                         if temp2(31)='1' and temp1(31)='1' and r1(127)='0' then
168                             alu_out(127 downto 96)<=(others=>'1');
169                             alu_out(127)<='0';
170                         elsif temp2(31)='0' and temp1(31)='0' and r1(127)='1' then
171                             alu_out(127 downto 96)<=(others=>'0');
172                             alu_out(127)<='1';
173                         else
174                             alu_out(127 downto 96)<=temp2;
175                         end if;
176
177                         temp1:=std_logic_vector(signed(r2(79 downto 64))* signed(r3(79
    downto 64)));
178                         temp2:=std_logic_vector(signed(r1(95 downto 64)) - signed(temp1));
179                         if temp2(31)='1' and temp1(31)='1' and r1(95)='0' then
180                             alu_out(95 downto 64)<=(others=>'1');
181                             alu_out(95)<='0';
182                         elsif temp2(31)='0' and temp1(31)='0' and r1(95)='1' then
183                             alu_out(95 downto 64)<=(others=>'0');
184                             alu_out(95)<='1';
185                         else
```

```
186                         alu_out(95 downto 64)<=temp2;
187                     end if;
188
189                     temp1:=std_logic_vector(signed(r2(47 downto 32))* signed(r3(47
    downto 32)));
190                     temp2:=std_logic_vector(signed(r1(63 downto 32)) - signed(temp1));
191                     if temp2(31)='1' and temp1(31)='1' and r1(63)='0' then
192                         alu_out(63 downto 32)<=(others=>'1');
193                         alu_out(63)<='0';
194                     elsif temp2(31)='0' and temp1(31)='0' and r1(63)='1' then
195                         alu_out(63 downto 32)<=(others=>'0');
196                         alu_out(63)<='1';
197                     else
198                         alu_out(63 downto 32)<=temp2;
199                     end if;
200
201                     temp1:=std_logic_vector(signed(r2(15 downto 0))* signed(r3(15 downto
    0)));
202                     temp2:=std_logic_vector(signed(r1(31 downto 0)) - signed(temp1));
203                     if temp2(31)='1' and temp1(31)='1' and r1(31)='0' then
204                         alu_out(31 downto 0)<=(others=>'1');
205                         alu_out(31)<='0';
206                     elsif temp2(31)='0' and temp1(31)='0' and r1(31)='1' then
207                         alu_out(31 downto 0)<=(others=>'0');
208                         alu_out(31)<='1';
209                     else
210                         alu_out(31 downto 0)<=temp2;
211                     end if;
212
213                     ----------Signed Integer Multiply-Subtract High with
    Saturation-------------
214                     when "011" =>
215                     temp1:=std_logic_vector(signed(r2(127 downto 112))* signed(r3(127
    downto 112)));
216                     temp2:=std_logic_vector(signed(r1(127 downto 96)) - signed(temp1));
217                     if temp2(31)='1' and temp1(31)='1' and r1(127)='0' then
218                         alu_out(127 downto 96)<=(others=>'1');
219                         alu_out(127)<='0';
220                     elsif temp2(31)='0' and temp1(31)='0' and r1(127)='1' then
221                         alu_out(127 downto 96)<=(others=>'0');
222                         alu_out(127)<='1';
223                     else
224                         alu_out(127 downto 96)<=temp2;
225                     end if;
226
227                     temp1:=std_logic_vector(signed(r2(95 downto 80))* signed(r3(95
    downto 80)));
228                     temp2:=std_logic_vector(signed(r1(95 downto 64)) - signed(temp1));
229                     if temp2(31)='1' and temp1(31)='1' and r1(95)='0' then
230                         alu_out(95 downto 64)<=(others=>'1');
231                         alu_out(95)<='0';
232                     elsif temp2(31)='0' and temp1(31)='0' and r1(95)='1' then
233                         alu_out(95 downto 64)<=(others=>'0');
234                         alu_out(95)<='1';
235                     else
236                         alu_out(95 downto 64)<=temp2;
237                     end if;
238
239                     temp1:=std_logic_vector(signed(r2(63 downto 48))* signed(r3(63
    downto 48)));
240                     temp2:=std_logic_vector(signed(r1(63 downto 32)) - signed(temp1));
241                     if temp2(31)='1' and temp1(31)='1' and r1(63)='0' then
242                         alu_out(63 downto 32)<=(others=>'1');
243                         alu_out(63)<='0';
244                     elsif temp2(31)='0' and temp1(31)='0' and r1(63)='1' then
245                         alu_out(63 downto 32)<=(others=>'0');
```

```
246                         alu_out(63)<='1';
247                     else
248                         alu_out(63 downto 32)<=temp2;
249                     end if;
250
251                     temp1:=std_logic_vector(signed(r2(31 downto 16))* signed(r3(31
    downto 16)));
252                     temp2:=std_logic_vector(signed(r1(31 downto 0)) - signed(temp1));
253                     if temp2(31)='1' and temp1(31)='1' and r1(31)='0' then
254                         alu_out(31 downto 0)<=(others=>'1');
255                         alu_out(31)<='0';
256                     elsif temp2(31)='0' and temp1(31)='0' and r1(31)='1' then
257                         alu_out(31 downto 0)<=(others=>'0');
258                         alu_out(31)<='1';
259                     else
260                         alu_out(31 downto 0)<=temp2;
261                     end if;
262
263                     ---------Signed Long Integer Multiply-Add Low with
    Saturation--------------
264                     when "100" =>
265                     temp3:=std_logic_vector(signed(r2(95 downto 64))* signed(r3(95
    downto 64)));
266                     temp4:=std_logic_vector(signed(temp3) + signed(r1(127 downto 64)));
267                     if temp4(63)='1' and temp3(63)='0' and r1(127)='0' then
268                         alu_out(127 downto 64)<=(others=>'1');
269                         alu_out(127)<='0';
270                     elsif temp4(63)='0' and temp3(63)='1' and r1(127)='1' then
271                         alu_out(127 downto 64)<=(others=>'0');
272                         alu_out(127)<='1';
273                     else
274                         alu_out(127 downto 64)<=temp4;
275                     end if;
276
277                     temp3:=std_logic_vector(signed(r2(31 downto 0))* signed(r3(31 downto
    0)));
278                     temp4:=std_logic_vector(signed(temp3) + signed(r1(63 downto 0)));
279                     if temp4(63)='1' and temp3(63)='0' and r1(63)='0' then
280                         alu_out(63 downto 0)<=(others=>'1');
281                         alu_out(63)<='0';
282                     elsif temp4(63)='0' and temp3(63)='1' and r1(63)='1' then
283                         alu_out(63 downto 0)<=(others=>'0');
284                         alu_out(63)<='1';
285                     else
286                         alu_out(63 downto 0)<=temp4;
287                     end if;
288                     ----------Signed Long Integer Multiply-Add High with
    Saturation-----------
289                     when "101" =>
290                         temp3:=std_logic_vector(signed(r2(127 downto 96))* signed(r3(127
    downto 96)));
291                     temp4:=std_logic_vector(signed(temp3) + signed(r1(127 downto 64)));
292                     if temp4(63)='1' and temp3(63)='0' and r1(127)='0' then
293                         alu_out(127 downto 64)<=(others=>'1');
294                         alu_out(127)<='0';
295                     elsif temp4(63)='0' and temp3(63)='1' and r1(127)='1' then
296                         alu_out(127 downto 64)<=(others=>'0');
297                         alu_out(127)<='1';
298                     else
299                         alu_out(127 downto 64)<=temp4;
300                     end if;
301
302                     temp3:=std_logic_vector(signed(r2(63 downto 32))* signed(r3(63
    downto 32)));
303                     temp4:=std_logic_vector(signed(temp3) + signed(r1(63 downto 0)));
304                     if temp4(63)='1' and temp3(63)='0' and r1(63)='0' then
```

```
305                         alu_out(63 downto 0)<=(others=>'1');
306                         alu_out(63)<='0';
307                     elsif temp4(63)='0' and temp3(63)='1' and r1(63)='1' then
308                         alu_out(63 downto 0)<=(others=>'0');
309                         alu_out(63)<='1';
310                     else
311                         alu out(63 downto 0)<=temp4;
312                     end if;
313                     ----------Signed Long Integer Multiply-Subtract Low with
    Saturation----------
314                     when "110" =>
315                     temp3:=std_logic_vector(signed(r2(95 downto 64))* signed(r3(95
    downto 64)));
316                     temp4:=std_logic_vector(signed(r1(127 downto 64))- signed(temp3));
317                     if temp4(63)='1' and temp3(63)='1' and r1(127)='0' then
318                         alu_out(127 downto 64)<=(others=>'1');
319                         alu_out(127)<='0';
320                     elsif temp4(63)='0' and temp3(63)='0' and r1(127)='1' then
321                         alu out(127 downto 64)<=(others=>'0');
322                         alu out(127)<='1';
323                     else
324                         alu out(127 downto 64)<=temp4;
325                     end if;
326
327                     temp3:=std_logic_vector(signed(r2(31 downto 0))* signed(r3(31 downto
    0)));
328                     temp4:=std_logic_vector(signed(r1(63 downto 0))-signed(temp3));
329                     if temp4(63)='1' and temp3(63)='1' and r1(63)='0' then
330                         alu_out(63 downto 0)<=(others=>'1');
331                         alu_out(63)<='0';
332                     elsif temp4(63)='0' and temp3(63)='0' and r1(63)='1' then
333                         alu out(63 downto 0)<=(others=>'0');
334                         alu out(63)<='1';
335                     else
336                         alu out(63 downto 0)<=temp4;
337                     end if;
338                     ----------Signed Long Integer Multiply-Subtract High with
    Saturation----------
339                     when "111" =>
340                     temp3:=std_logic_vector(signed(r2(127 downto 96))* signed(r3(127
    downto 96)));
341                     temp4:=std_logic_vector(signed(r1(127 downto 64))- signed(temp3));
342                     if temp4(63)='1' and temp3(63)='1' and r1(127)='0' then
343                         alu out(127 downto 64)<=(others=>'1');
344                         alu out(127)<='0';
345                     elsif temp4(63)='0' and temp3(63)='0' and r1(127)='1' then
346                         alu out(127 downto 64)<=(others=>'0');
347                         alu out(127)<='1';
348                     else
349                         alu_out(127 downto 64)<=temp4;
350                     end if;
351
352                     temp3:=std_logic_vector(signed(r2(63 downto 32))* signed(r3(63
    downto 32)));
353                     temp4:=std_logic_vector(signed(r1(63 downto 0))-signed(temp3));
354                     if temp4(63)='1' and temp3(63)='0' and r1(63)='1' then
355                         alu out(63 downto 0)<=(others=>'1');
356                         alu out(63)<='0';
357                     elsif temp4(63)='0' and temp3(63)='1' and r1(63)='0' then
358                         alu out(63 downto 0)<=(others=>'0');
359                         alu out(63)<='1';
360                     else
361                         alu_out(63 downto 0)<=temp4;
362                     end if;
363                     when others  => null;
364
```

```
365                 end case;
366
367
368         ----------------4.3 R3 instruction-----------------------------------
369             elsif ins(24 downto 23) = "11" then
370             case ins(19 downto 15) is
371                 ---------NOP---------
372                 when "00000" =>  null;
373                 --------A: add word---------
374                 when "00001" =>
375                     alu_out(127 downto 96)<=std_logic_vector(unsigned(r2(127 downto 96
    ))+unsigned(r1(127 downto 96)));
376                     alu_out(95 downto 64)<=std_logic_vector(unsigned(r2(95 downto 64))+
    unsigned(r1(95 downto 64)));
377                     alu_out(63 downto 32)<=std_logic_vector(unsigned(r2(63 downto 32))+
    unsigned(r1(63 downto 32)));
378                     alu_out(31 downto 0)<=std_logic_vector(unsigned(r2(31 downto 0))+
    unsigned(r1(31 downto 0)));
379                 ----AH: add halfword--
380                 when "00010" =>
381                     alu_out(127 downto 112)<=std_logic_vector(unsigned(r2(127 downto 112
    ))+unsigned(r1(127 downto 112)));
382                     alu_out(111 downto 96)<=std_logic_vector(unsigned(r2(111 downto 96
    ))+unsigned(r1(111 downto 96)));
383                     alu_out(95 downto 80)<=std_logic_vector(unsigned(r2(95 downto 80))+
    unsigned(r1(95 downto 80)));
384                     alu_out(79 downto 64)<=std_logic_vector(unsigned(r2(79 downto 64))+
    unsigned(r1(79 downto 64)));
385                     alu_out(63 downto 48)<=std_logic_vector(unsigned(r2(63 downto 48))+
    unsigned(r1(63 downto 48)));
386                     alu_out(47 downto 32)<=std_logic_vector(unsigned(r2(47 downto 32))+
    unsigned(r1(47 downto 32)));
387                     alu_out(31 downto 16)<=std_logic_vector(unsigned(r2(31 downto 16))+
    unsigned(r1(31 downto 16)));
388                     alu_out(15 downto 0)<=std_logic_vector(unsigned(r2(15 downto 0))+
    unsigned(r1(15 downto 0)));
389                 ----AHS: add halfword saturated
390                 when "00011" =>
391                 alu_out(127 downto 112)<=std_logic_vector(signed(r2(127 downto 112)) +
    signed(r1(127 downto 112)));
392                 if alu_out(127)='1' and r2(127) = '0' and r1(127) = '0' then

393                     alu_out(127 downto 112)<=(others=>'1');
394                     alu_out(127)<='0';
395                 elsif alu_out(127) ='0' and r2(127) ='1' and r1(127)='1' then
396                     alu_out(127 downto 112)<=(others=>'0');
397                     alu_out(127)<='1';
398                 end if;
399                 alu_out(111 downto 96)<=std_logic_vector(signed(r2(111 downto 96))+
    signed(r1(111 downto 96)));
400                 if alu_out(111)='1' and r2(111) = '0' and r1(111) = '0' then

401                     alu_out(111 downto 96)<=(others=>'1');
402                     alu_out(111)<='0';
403                 elsif alu_out(111) ='0' and r2(111) ='1' and r1(111)='1' then
404                     alu_out(111 downto 96)<=(others=>'0');
405                     alu_out(111)<='1';
406                 end if;
407                 alu_out(95 downto 80)<=std_logic_vector(signed(r2(95 downto 80))+signed(
    r1(95 downto 80)));
408                 if alu_out(95)='1' and r2(95) = '0' and r1(95) = '0' then

409                     alu_out(95 downto 80)<=(others=>'1');
410                     alu_out(95)<='0';
411                 elsif alu_out(95) ='0' and r2(95) ='1' and r1(95)='1' then
412                     alu_out(95 downto 80)<=(others=>'0');
```

```
413                     alu_out(95)<='1';
414                 end if;
415                 alu_out(79 downto 64)<=std_logic_vector(signed(r2(79 downto 64))+signed(
    r1(79 downto 64)));
416                 if alu_out(79)='1' and r2(79) = '0' and r1(79) = '0' then

417                     alu out(79 downto 64)<=(others=>'1');
418                     alu out(79)<='0';
419                 elsif alu out(79) ='0' and r2(79) ='1' and r1(79)='1' then
420                     alu out(79 downto 64)<=(others=>'0');
421                     alu out(79)<='1';
422                 end if;
423                 alu_out(63 downto 48)<=std_logic_vector(signed(r2(63 downto 48))+signed(
    r1(63 downto 48)));
424                 if alu_out(63)='1' and r2(63) = '0' and r1(63) = '0' then

425                     alu_out(63 downto 48)<=(others=>'1');
426                     alu_out(63)<='0';
427                 elsif alu out(63) ='0' and r2(63) ='1' and r1(63)='1' then
428                     alu out(63 downto 48)<=(others=>'0');
429                     alu out(63)<='1';
430                 end if;
431                 alu out(47 downto 32)<=std_logic_vector(signed(r2(47 downto 32))+signed(
    r1(47 downto 32)));
432                 if alu_out(47)='1' and r2(47) = '0' and r1(32) = '0' then

433                     alu_out(47 downto 32)<=(others=>'1');
434                     alu_out(47)<='0';
435                 elsif alu_out(47) ='0' and r2(47) ='1' and r1(32)='1' then
436                     alu_out(47 downto 32)<=(others=>'0');
437                     alu out(47)<='1';
438                 end if;
439                 alu out(31 downto 16)<=std_logic_vector(signed(r2(31 downto 16))+signed(
    r1(31 downto 16)));
440                 if alu out(31)='1' and r2(31) = '0' and r1(31) = '0' then

441                     alu out(31 downto 16)<=(others=>'1');
442                     alu_out(31)<='0';
443                 elsif alu_out(31) ='0' and r2(31) ='1' and r1(31)='1' then
444                     alu_out(31 downto 16)<=(others=>'0');
445                     alu_out(31)<='1';
446                 end if;
447                 alu_out(15 downto 0)<=std_logic_vector(signed(r2(15 downto 0))+signed(r1
    (15 downto 0)));
448                 if alu out(15)='1' and r2(15) = '0' and r1(15) = '0' then

449                     alu out(15 downto 0)<=(others=>'1');
450                     alu out(15)<='0';
451                 elsif alu out(15) ='0' and r2(15) ='1' and r1(15)='1' then
452                     alu_out(15 downto 0)<=(others=>'0');
453                     alu_out(15)<='1';
454                 end if;
455                 --------AND: bitwise logical and
456                 when "00100" =>
457                 alu_out <= r1 and r2;
458                 --------BCW: broadcast word
459                 when "00101" =>
460                 alu out(127 downto 96)<=r1(31 downto 0);
461                 alu out(95 downto 64)<=r1(31 downto 0);
462                 alu out(63 downto 32)<=r1(31 downto 0);
463                 alu out(31 downto 0)<=r1(31 downto 0);
464
465                 -------CLZ: count leading zeros in words
466                 when "00110" =>
467                 zero_count := 0;
468                     for i in 127 downto 96 loop
```

```
469                         if r1(i) = '0' then
470                             zero_count := zero_count+1;
471                         else
472                             exit;
473                         end if;
474                     end loop;
475                     alu_out(127 downto 96) <= std_logic_vector(to_unsigned(zero_count,32
    ));
476                     zero_count :=0;
477
478                     for i in 95 downto 64 loop
479                         if r1(i) = '0' then
480                             zero_count := zero_count+1;
481                         else
482                             exit;
483                         end if;
484                     end loop;
485                     alu_out(95 downto 64) <= std_logic_vector(to_unsigned(zero_count,32
    ));
486                     zero_count :=0;
487
488                     for i in 63 downto 32 loop
489                         if r1(i) = '0' then
490                             zero_count := zero_count+1;
491                         else
492                             exit;
493                         end if;
494                     end loop;
495                     alu_out(63 downto 32) <= std_logic_vector(to_unsigned(zero_count,32
    ));
496                     zero_count :=0;
497
498                     for i in 31 downto 0 loop
499                         if r1(i) = '0' then
500                             zero_count := zero_count+1;
501                         else
502                             exit;
503                         end if;
504                     end loop;
505                     alu_out(31 downto 0) <= std_logic_vector(to_unsigned(zero_count,32
    ));
506                     zero_count :=0;
507                 ---------MAX: max signed word
508                 when "00111" =>
509                     if signed(r1(127 downto 96)) > signed(r2(127 downto 96)) then
510                         alu_out(127 downto 96)<= r1(127 downto 96);
511                         else
512                             alu_out(127 downto 96) <= r2(127 downto 96);
513                     end if;
514
515                     if(signed(r1(95 downto 64)) > signed(r2(95 downto 64))) then
516                         alu_out(95 downto 64)<= r1(95 downto 64);
517                         else
518                             alu_out(95 downto 64) <= r2(95 downto 64);
519                     end if;
520
521                     if(signed(r1(63 downto 32)) > signed(r2(63 downto 32))) then
522                         alu_out(63 downto 32)<= r1(63 downto 32);
523                         else
524                             alu_out(63 downto 32) <= r2(63 downto 32);
525                     end if;
526
527                     if(signed(r1(31 downto 0)) > signed(r2(31 downto 0))) then
528                         alu_out(31 downto 0)<= r1(31 downto 0);
529                         else
530                             alu_out(31 downto 0) <= r2(31 downto 0);
```

```
531                     end if;
532
533
534                 ----------MIN: min signed word
535                 when "01000" =>
536                     if (to integer(signed(r1(127 downto 96))) > to integer(signed(r2(127
    downto 96)))) then
537                         alu out(127 downto 96)<= r2(127 downto 96);
538                         else
539                             alu out(127 downto 96) <= r1(127 downto 96);
540                     end if;
541
542                     if signed(r1(95 downto 64)) > signed(r2(95 downto 64)) then
543                         alu_out(95 downto 64)<= r2(95 downto 64);
544                         else
545                             alu_out(95 downto 64) <= r1(95 downto 64);
546                     end if;
547
548                     if signed(r1(63 downto 32)) > signed(r2(63 downto 32)) then
549                         alu out(63 downto 32)<= r2(63 downto 32);
550                         else
551                             alu out(63 downto 32) <= r1(63 downto 32);
552                     end if;
553
554                     if signed(r1(31 downto 0)) > signed(r2(31 downto 0)) then
555                         alu_out(31 downto 0)<= r2(31 downto 0);
556                         else
557                             alu_out(31 downto 0) <= r1(31 downto 0);
558                     end if;
559
560
561                 ---------MSGN: multiply signed
562                 -----???
563                 when "01001" =>
564                 if to integer(signed(r1(127 downto 96)))= -2147483648 and r2(127)='1'
    then
565                     alu out(127 downto 96) <= (Others=>'1');
566                     alu_out(127)<='0';
567                 else
568                     if r2(127)='1' then
569                         --signed(r1(127 downto 96)*(-1)
570                         alu_out(127 downto 96)<=std_logic_vector(to_signed(to_integer(
    signed(r1(127 downto 96)))*(-1),32));
571
572                     elsif r2(127 downto 96)="00000000000000000000000000000000" then
573                         alu out(127 downto 96)<="00000000000000000000000000000000" ;
574                     else
575                         alu out(127 downto 96)<=r1(127 downto 96);
576                     end if;
577                 end if;
578
579
580                 if to_integer(signed(r1(95 downto 64)))= -2147483648 and (r2(127)='1')
    then
581                     alu_out(95 downto 64)<=(Others=>'1');
582                     alu out(95)<='0';
583                 else
584                     if(r2(95)='1') then
585                         alu out(95 downto 64)<=std_logic_vector(to signed(to integer(
    signed(r1(95 downto 64)))*(-1),32));
586                         elsif unsigned(r2(95 downto 64))=0 then
587                         alu out(95 downto 64)<="00000000000000000000000000000000" ;
588                         else
589                         alu_out(95 downto 64)<=r1(95 downto 64);
590                     end if;
591                 end if;
```

```
592
593                 if to_integer(signed(r1(63 downto 32)))= -2147483648 and (r2(127)='1')
    then
594                     alu_out(63 downto 32)<=(Others=>'1');
595                     alu_out(63)<='0';
596                 else
597                     if(r2(63)='1') then
598                         alu out(63 downto 32)<=std_logic_vector(to signed(to integer(
    signed(r1(63 downto 32)))*(-1),32));
599                         elsif unsigned(r2(63 downto 32))=0 then
600                         alu out(63 downto 32)<="00000000000000000000000000000000" ;
601                         else
602                         alu_out(63 downto 32)<=r1(63 downto 32);
603                     end if;
604                 end if;
605
606
607                 if to_integer(signed(r1(31 downto 0)))= -2147483648 and (r2(127)='1')
    then
608                     alu out(31 downto 0)<=(Others=>'1');
609                     alu out(31)<='0';
610                 else
611                     if(r2(31)='1') then
612                         alu out(31 downto 0)<=std_logic_vector(to signed(to integer(
    signed(r1(31 downto 0)))*(-1),32));
613                         elsif unsigned(r2(31 downto 0))=0 then
614                         alu_out(31 downto 0)<="00000000000000000000000000000000" ;
615                         else
616                         alu_out(31 downto 0)<=r1(31 downto 0);
617                     end if;
618                 end if;
619                 ---------MPYU: multiply unsigned
620                 when "01010" =>
621                 alu out(127 downto 96)<=std_logic_vector(to unsigned(to integer(unsigned
    (r1(111 downto 96)))*to integer( unsigned(r2(111 downto 96))),32));
622                 alu out(95 downto 64)<=std_logic_vector(to unsigned(to integer(unsigned(
    r1(79 downto 64)))*to integer( unsigned(r2(79 downto 64))),32));
623                 alu_out(63 downto 32)<=std_logic_vector(to_unsigned(to_integer(unsigned(
    r1(47 downto 32)))*to_integer( unsigned(r2(47 downto 32))),32));
624                 alu_out(31 downto 0)<=std_logic_vector(to_unsigned(to_integer(unsigned(
    r1(15 downto 0)))*to_integer( unsigned(r2(15 downto 0))),32));
625                 ---------OR: bitwise logical or
626                 when "01011" =>
627                     alu out <= r1 or r2;
628
629                 ---------POPCNTH: count ones in halfwords
630                 when "01100" =>
631                 one count := 0;
632                     for i in 127 downto 112 loop
633                         if r1(i)='1' then
634                             one_count := one_count+1;
635                         end if;
636                     end loop;
637                     alu_out(127 downto 112) <= std_logic_vector(to_unsigned(one_count,16
    ));
638                     one count:=0;
639
640                     for i in 111 downto 96 loop
641                         if r1(i)='1' then
642                             one count := one count +1;
643                         end if;
644                     end loop;
645                     alu_out(111 downto 96) <= std_logic_vector(to_unsigned(one_count,16
    ));
646                     one_count:=0;
647
```

```
648                     for i in 95 downto 80 loop
649                         if r1(i)='1' then
650                             one_count := one_count+1;
651                         end if;
652                     end loop;
653                     alu_out(95 downto 80) <= std_logic_vector(to_unsigned(one_count,16
    ));
654                     one_count:=0;
655
656                     for i in 79 downto 64 loop
657                         if r1(i)='1' then
658                             one_count := one_count +1;
659                         end if;
660                     end loop;
661                     alu_out(79 downto 64) <= std_logic_vector(to_unsigned(one_count,16
    ));
662                     one_count:=0;
663
664                     for i in 63 downto 48 loop
665                         if r1(i)='1' then
666                             one_count := one_count+1;
667                         end if;
668                     end loop;
669                     alu_out(63 downto 48) <= std_logic_vector(to_unsigned(one_count,16
    ));
670                     one_count:=0;
671
672                     for i in 47 downto 32 loop
673                         if r1(i)='1' then
674                             one_count := one_count +1;
675                         end if;
676                     end loop;
677                     alu_out(47 downto 32) <= std_logic_vector(to_unsigned(one_count,16
    ));
678                     one_count:=0;
679
680                     for i in 31 downto 16 loop
681                         if r1(i)='1' then
682                             one_count := one_count+1;
683                         end if;
684                     end loop;
685                     alu_out(31 downto 16) <= std_logic_vector(to_unsigned(one_count,16
    ));
686                     one_count:=0;
687
688                     for i in 15 downto 0 loop
689                         if r1(i)='1' then
690                             one_count := one_count +1;
691                         end if;
692                     end loop;
693                     alu_out(15 downto 0) <= std_logic_vector(to_unsigned(one_count,16));
694                     one_count:=0;
695
696                 ---------ROT: rotate bits right
697                 when "01101" =>
698                     bits_of_shift:=to_integer(unsigned(r2(6 downto 0)));
699                     alu_out<=r1 ror bits_of_shift;
700
701                 ---------ROTW: rotate bits in word
702                 when "01110" =>
703                     bits_of_shift:=to_integer(unsigned(r2(100 downto 96)));
704                     alu_out(127 downto 96)<=r1(127 downto 96) ror bits_of_shift;
705                     bits_of_shift:=to_integer(unsigned(r2(68 downto 64)));
706                     alu_out(95 downto 64)<=r1(95 downto 64) ror bits_of_shift;
707                     bits_of_shift:=to_integer(unsigned(r2(36 downto 32)));
708                     alu_out(63 downto 32)<=r1(63 downto 32) ror bits_of_shift;
```

```
709                     bits_of_shift:=to_integer(unsigned(r2(4 downto 0)));
710                     alu_out(31 downto 0)<=r1(31 downto 0) ror bits_of_shift;
711
712                 ---------SHLHI: shift left halfword immediate:
713                 ------???
714                 when "01111" =>
715                     bits of shift:=to integer(unsigned(r2(115 downto 112)));
716                     alu out(127 downto 112)<=r1(127 downto 112) SLL bits of shift;
717                     bits of shift:=to integer(unsigned(r2(99 downto 96)));
718                     alu out(111 downto 96)<=r1(111 downto 96) SLL bits of shift;
719                     bits of shift:=to integer(unsigned(r2(83 downto 80)));
720                     alu out(95 downto 80)<=r1(95 downto 80) SLL bits of shift;
721                     bits_of_shift:=to_integer(unsigned(r2(67 downto 64)));
722                     alu_out(79 downto 64)<=r1(79 downto 64) SLL bits_of_shift;
723                     bits_of_shift:=to_integer(unsigned(r2(51 downto 48)));
724                     alu_out(63 downto 48)<=r1(63 downto 48) SLL bits_of_shift;
725                     bits_of_shift:=to_integer(unsigned(r2(35 downto 32)));
726                     alu_out(47 downto 32)<=r1(47 downto 32) SLL bits_of_shift;
727                     bits of shift:=to integer(unsigned(r2(19 downto 16)));
728                     alu out(31 downto 16)<=r1(31 downto 16) SLL bits of shift;
729                     bits of shift:=to integer(unsigned(r2(3 downto 0)));
730                     alu out(15 downto 0)<=r1(15 downto 0) SLL bits of shift;
731
732                 ---------SFH: subtract from halfword immediate
733                 when "10000" =>
734                 alu_out(127 downto 112)<=std_logic_vector(to_unsigned(to_integer(
    unsigned(r2(127 downto 112)))-to_integer(unsigned(r2(127 downto 112))),16));
735                 alu_out(111 downto 96)<=std_logic_vector(to_unsigned(to_integer(unsigned
    (r2(111 downto 96)))-to_integer(unsigned(r2(111 downto 96))),16));
736                 alu_out(95 downto 80)<=std_logic_vector(to_unsigned(to_integer(unsigned(
    r2(95 downto 80)))-to integer(unsigned(r2(95 downto 80))),16));
737                 alu out(79 downto 64)<=std_logic_vector(to unsigned(to integer(unsigned(
    r2(79 downto 64)))-to integer(unsigned(r2(79 downto 64))),16));
738                 alu out(63 downto 48)<=std_logic_vector(to unsigned(to integer(unsigned(
    r2(63 downto 48)))-to integer(unsigned(r2(63 downto 48))),16));
739                 alu out(47 downto 32)<=std_logic_vector(to unsigned(to integer(unsigned(
    r2(47 downto 32)))-to integer(unsigned(r2(47 downto 32))),16));
740                 alu_out(31 downto 16)<=std_logic_vector(to_unsigned(to_integer(unsigned(
    r2(31 downto 16)))-to_integer(unsigned(r2(31 downto 16))),16));
741                 alu_out(15 downto 0)<=std_logic_vector(to_unsigned(to_integer(unsigned(
    r2(15 downto 0)))-to_integer(unsigned(r2(15 downto 0))),16));
742
743                 ---------SFW: subtract from word
744                 when "10001" =>
745                 alu out(127 downto 96)<=std_logic_vector(to unsigned(to integer(unsigned
    (r2(127 downto 96)))-to integer(unsigned(r2(127 downto 96))),32));
746                 alu out(95 downto 64)<=std_logic_vector(to unsigned(to integer(unsigned(
    r2(95 downto 64)))-to integer(unsigned(r2(95 downto 64))),32));
747                 alu out(63 downto 32)<=std_logic_vector(to unsigned(to integer(unsigned(
    r2(63 downto 32)))-to_integer(unsigned(r2(63 downto 32))),32));
748                 alu_out(31 downto 0)<=std_logic_vector(to_unsigned(to_integer(unsigned(
    r2(31 downto 0)))-to_integer(unsigned(r2(31 downto 0))),32));
749
750                 ---------SFHS: subtract from halfword saturated
751                 ---------saturation??
752                 when "10010" =>
753                 alu out(127 downto 112)<=std_logic_vector(signed(r2(127 downto 112)) -
    signed(r1(127 downto 112)));
754                 if alu out(127)='0' and r2(127)='1' and r1(127)='0'  then
755                     alu out(127 downto 112) <= (others=>'0');
756                     alu out(127)<='1';
757                 elsif alu out(127)='1' and r2(127)='0' and r2(127)='1' then
758                     alu_out(127 downto 112) <= (others=>'1');
759                     alu_out(127)<='0';
760                 end if;
761
```

```
762                     alu_out(111 downto 96)<=std_logic_vector(signed(r2(111 downto 96))-
    signed(r1(111 downto 96)));
763                     if alu_out(111)='0' and r2(111)='1' and r1(111)='0'  then
764                         alu_out(111 downto 96) <= (others=>'0');
765                         alu_out(111)<='1';
766                     elsif alu out(111)='1' and r2(111)='0' and r2(111)='1' then
767                         alu out(111 downto 96) <= (others=>'1');
768                         alu out(111)<='0';
769                     end if;
770
771                     alu out(95 downto 80)<=std_logic_vector(signed(r2(95 downto 80))-signed(
    r1(95 downto 80)));
772                     if alu_out(95)='0' and r2(95)='1' and r1(95)='0'  then
773                         alu_out(95 downto 80) <= (others=>'0');
774                         alu_out(95)<='1';
775                     elsif alu_out(95)='1' and r2(95)='0' and r2(95)='1' then
776                         alu_out(95 downto 80) <= (others=>'1');
777                         alu_out(95)<='0';
778                     end if;
779
780                     alu out(79 downto 64)<=std_logic_vector(signed(r2(79 downto 64))-signed(
    r1(79 downto 64)));
781                     if alu out(79)='0' and r2(79)='1' and r1(79)='0'  then
782                         alu out(79 downto 64) <= (others=>'0');
783                         alu_out(79)<='1';
784                     elsif alu_out(79)='1' and r2(79)='0' and r2(79)='1' then
785                         alu_out(79 downto 64) <= (others=>'1');
786                         alu_out(79)<='0';
787                     end if;
788
789                     alu out(63 downto 48)<=std_logic_vector(signed(r2(63 downto 48))-signed(
    r1(63 downto 48)));
790                     if alu out(63)='0' and r2(63)='1' and r1(63)='0'  then
791                         alu out(63 downto 48) <= (others=>'0');
792                         alu out(63)<='1';
793                     elsif alu out(63)='1' and r2(63)='0' and r2(63)='1' then
794                         alu out(63 downto 48) <= (others=>'1');
795                         alu_out(63)<='0';
796                     end if;
797
798                     alu_out(47 downto 32)<=std_logic_vector(signed(r2(47 downto 32))-signed(
    r1(47 downto 32)));
799                     if alu_out(47)='0' and r2(47)='1' and r1(47)='0'  then
800                         alu out(47 downto 32) <= (others=>'0');
801                         alu out(47)<='1';
802                     elsif alu out(47)='1' and r2(47)='0' and r2(47)='1' then
803                         alu out(47 downto 32) <= (others=>'1');
804                         alu out(47)<='0';
805                     end if;
806
807                     alu_out(31 downto 16)<=std_logic_vector(signed(r2(31 downto 16))-signed(
    r1(31 downto 16)));
808                     if alu_out(31)='0' and r2(31)='1' and r1(31)='0'  then
809                         alu_out(31 downto 16) <= (others=>'0');
810                         alu_out(31)<='1';
811                     elsif alu out(31)='1' and r2(31)='0' and r2(31)='1' then
812                         alu out(31 downto 16) <= (others=>'1');
813                         alu out(31)<='0';
814                     end if;
815
816                     alu out(15 downto 0)<=std_logic_vector(signed(r2(15 downto 0))-signed(r1
    (15 downto 0)));
817                     if alu_out(15)='0' and r2(15)='1' and r1(15)='0'  then
818                         alu_out(15 downto 0) <= (others=>'0');
819                         alu_out(15)<='1';
820                     elsif alu_out(15)='1' and r2(15)='0' and r2(15)='1' then
```

```
821                     alu_out(15 downto 0) <= (others=>'1');
822                     alu_out(15)<='0';
823                 end if;
824
825                 ---------XOR: bitwise logical exclusive-or
826                 when "10011" =>
827                 alu out <= r1 xor r2;
828                 when others => null;
829
830             end case;
831         end if;
832         end if;
833     end process;
834
835 end behavioral;
```

```
1     ----------------------------------------------------------------------------
2     --
3     -- Title       : forwarding_unit
4     -- Design      : processor
5     -- Author      :
6     -- Company     :
7     --
8     ----------------------------------------------------------------------------
9     --
10    -- File        :
      C:\Users\gavin\Desktop\Study\ESE345\project\processor\processor\src\forwarding_unit.vhd
11    -- Generated   : Sat Nov 30 17:16:23 2019
12    -- From        : interface description file
13    -- By          : Itf2Vhdl ver. 1.22
14    --
15    ----------------------------------------------------------------------------
16    --
17    -- Description :
18    --
19    ----------------------------------------------------------------------------
20
21    --{{ Section below this comment is automatically maintained
22    --   and may be overwritten
23    --{entity {forwarding_unit} architecture {\behavioral \}}
24
25    library ieee;
26    use ieee.std_logic_1164.all;
27    use ieee.numeric_std.all;
28
29    entity forwarding_unit is
30        port(
31        ALU_out : in std_logic_vector(127 downto 0);
32        r1_in : in std_logic_vector(127 downto 0);
33        r2_in : in std_logic_vector(127 downto 0);
34        r3_in : in std_logic_vector(127 downto 0);
35        ins : in std_logic_vector(24 downto 0);
36        old_ins : in std_logic_vector(24 downto 0);
37        r1_out : out std_logic_vector(127 downto 0);
38        r2_out : out std_logic_vector(127 downto 0);
39        r3_out : out std_logic_vector(127 downto 0)
40        );
41    end forwarding_unit;
42
43    --}} End of automatically maintained section
44
45    architecture behavioral of forwarding_unit is
46    signal older_ins : std_logic_vector(24 downto 0);
47    signal old_out : std_logic_vector(127 downto 0);
48    begin
49        process(ALU_out, r1_in, r2_in, r3_in, ins, old_ins)
50        begin
51            if ins(24) = '0' then
52                if old_ins(4 downto 0) = ins(4 downto 0) then
53                    r1_out <= ALU_out;
54                    r2_out <= r2_in;
55                    r3_out <= r3_in;
56                elsif older_ins(4 downto 0) = ins(4 downto 0) then
57                    r1_out <= old_out;
58                    r2_out <= r2_in;
59                    r3_out <= r3_in;
60                else
61                    r1_out <= r1_in;
62                    r2_out <= r2_in;
63                    r3_out <= r3_in;
64                end if;
65            elsif ins(24) = '1' then
```

```
66              if ins(23) = '1' then
67                  if ins(19 downto 15) = "00000" then
68                      r1_out <= r1_in;
69                      r2_out <= r2_in;
70                      r3_out <= r3_in;
71                  else
72                      if old ins(4 downto 0) = ins(9 downto 5) then
73                          r1 out <= ALU out;
74                          r2 out <= r2 in;
75                          r3 out <= r3 in;
76                      elsif old ins(4 downto 0) = ins(14 downto 10) then
77                          r2 out <= ALU out;
78                          r1_out <= r1_in;
79                          r3_out <= r3_in;
80                      elsif older_ins(4 downto 0) = ins(9 downto 5) then
81                          r1_out <= old_out;
82                          r2_out <= r2_in;
83                          r3_out <= r3_in;
84                      elsif older ins(4 downto 0) = ins(14 downto 10) then
85                          r2 out <= old out;
86                          r1 out <= r1 in;
87                          r3 out <= r3 in;
88                      else
89                          r1 out <= r1 in;
90                          r2_out <= r2_in;
91                          r3_out <= r3_in;
92                      end if;
93                  end if;
94              elsif ins(23) = '0' then
95                  if old_ins(4 downto 0) = ins(9 downto 5) then
96                      r1 out <= ALU out;
97                      r2 out <= r2 in;
98                      r3 out <= r3 in;
99                  elsif old ins(4 downto 0) = ins(14 downto 10) then
100                     r2 out <= ALU out;
101                     r1 out <= r1 in;
102                     r3 out <= r3 in;
103                 elsif old_ins(4 downto 0) = ins(19 downto 15) then
104                     r3_out <= ALU_out;
105                     r1_out <= r1_in;
106                     r2_out <= r2_in;
107                 elsif older_ins(4 downto 0) = ins(9 downto 5) then
108                     r1_out <= old_out;
109                     r2 out <= r2 in;
110                     r3 out <= r3 in;
111                 elsif older ins(4 downto 0) = ins(14 downto 10) then
112                     r2 out <= old out;
113                     r1 out <= r1 in;
114                     r3 out <= r3 in;
115                 elsif older_ins(4 downto 0) = ins(19 downto 15) then
116                     r3_out <= old_out;
117                     r1_out <= r1_in;
118                     r2_out <= r2_in;
119                 else
120                     r1_out <= r1_in;
121                     r2 out <= r2 in;
122                     r3 out <= r3 in;
123                 end if;
124             end if;
125         end if;
126         older ins <= old ins;
127         old out <= ALU out;
128     end process;
129
130
131      -- enter your statements here --
```

File: f:/project345/src/forwarding_unit.vhd (/file_io_tb/UUT/u4)

```
132
133   end behavioral;
134
```

```
25  library ieee;
26  use ieee.std_logic_1164.all;
27  use ieee.numeric_std.all;
28
29  package insbuffer_type is
30  type insBuffer is array(63 downto 0) of std_logic_vector (24 downto 0);
31  end package insbuffer type;
32  use work.insbuffer type.all;
33
34  library ieee;
35  use ieee.std logic 1164.all;
36  use ieee.numeric std.all;
37  use IEEE.STD_LOGIC_UNSIGNED.ALL;
38
39  entity ins_buffer is
40      port(
41      instructions : in insBuffer;
42      clk : in std_logic;
43      ins out : out std_logic_vector(24 downto 0)
44      );
45  end ins buffer;
46
47  --}} End of automatically maintained section
48
49  architecture behavioral of ins_buffer is
50  signal PC : std_logic_vector(5 downto 0) := "000000";
51  begin
52      process(clk, instructions)
53      begin
54          if rising_edge(clk) then
55              ins out <= instructions(to integer(unsigned(PC)));
56              PC <= PC + std_logic_vector(to unsigned(1, PC'length));
57          end if;
58      end process;
59
60       -- enter your statements here --
61
62  end behavioral;
```

```
1   ----------------------------------------------------------------------------
2   --
3   -- Title       : register_file
4   -- Design      : processor
5   -- Author      :
6   -- Company     :
7   --
8   ----------------------------------------------------------------------------
9   --
10  -- File        :
    C:\Users\gavin\Desktop\Study\ESE345\project\processor\processor\src\register file.vhd
11  -- Generated   : Sat Nov 30 15:35:50 2019
12  -- From        : interface description file
13  -- By          : Itf2Vhdl ver. 1.22
14  --
15  ----------------------------------------------------------------------------
16  --
17  -- Description :
18  --
19  ----------------------------------------------------------------------------
20
21  --{{ Section below this comment is automatically maintained
22  --   and may be overwritten
23  --{entity {register file} architecture {behavioral}}
24
25  library IEEE;
26  use IEEE.std_logic_1164.all;
27  use ieee.numeric_std.all;
28
29  entity register_file is
30       port(
31            WrData : in STD_LOGIC_VECTOR(127 downto 0);
32            ins : in STD_LOGIC_VECTOR(24 downto 0);
33            old ins : in std_logic_vector(24 downto 0);
34            r1 : out STD_LOGIC_VECTOR(127 downto 0);
35            r2 : out STD_LOGIC_VECTOR(127 downto 0);
36            r3 : out STD_LOGIC_VECTOR(127 downto 0);
37            ins_out : out std_logic_vector(24 downto 0);
38            clk : in std_logic
39            );
40  end register_file;
41
42  --}} End of automatically maintained section
43
44  architecture behavioral of register file is
45  type RegisterFile is array(31 downto 0) of std_logic_vector(127 downto 0);
46  signal regs : RegisterFile := ((others=>(others=>'0')));

47  begin
48      process(clk, WrData, ins)
49      begin
50          if rising_edge(clk) then
51              if old_ins(24 downto 23) = "11" then
52                  if old_ins(19 downto 15) = "00000" then
53                  null;
54                  else
55                      regs(to integer(unsigned(old ins(4 downto 0)))) <= WrData;
56                  end if;
57              else
58                  regs(to integer(unsigned(old ins(4 downto 0)))) <= WrData;
59              end if;
60              if ins(24) = '0' then
61                  r1 <= regs(to_integer(unsigned(ins(4 downto 0))));
62                  r2 <= regs(to_integer(unsigned(ins(14 downto 10))));

63                  r3 <= regs(to_integer(unsigned(ins(19 downto 15))));
```

```
64              end if;
65              if ins(9 downto 5) = old_ins(4 downto 0) then
66                  r1 <= WrData;
67                  r2 <= regs(to_integer(unsigned(ins(14 downto 10))));

68                  r3 <= regs(to integer(unsigned(ins(19 downto 15))));
69              elsif ins(9 downto 5) = old ins(4 downto 0) then
70                  r2 <= WrData;
71                  r1 <= regs(to integer(unsigned(ins(9 downto 5))));

72                  r3 <= regs(to integer(unsigned(ins(19 downto 15))));
73              elsif ins(9 downto 5) = old ins(4 downto 0) then
74                  r3 <= WrData  ;
75                  r2 <= regs(to_integer(unsigned(ins(14 downto 10))));

76                  r1 <= regs(to_integer(unsigned(ins(9 downto 5))));
77              else
78                  r1 <= regs(to_integer(unsigned(ins(9 downto 5))));
79                  r2 <= regs(to integer(unsigned(ins(14 downto 10))));

80                  r3 <= regs(to integer(unsigned(ins(19 downto 15))));
81              end if;
82              ins out <= ins;
83          end if;
84      end process;
85
86       -- enter your statements here --
87
88  end behavioral;
89
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.numeric_std.all;
4   use STD.textio.all;
5   use ieee.std_logic_textio.all;
6   use work.insbuffer_type.all;
7
8   entity file_io_tb is
9   end file_io_tb;
10
11
12  architecture behave of file_io_tb is
13
14    ------------------------------------------------------------------------
    --
15    -- Declare the Component Under Test
16    ------------------------------------------------------------------------
    --
17
18    signal tb_clk: std_logic;
19    signal tb_alu_out: std_logic_vector(127 downto 0);
20    signal rom_data: insBuffer;
21    signal count:integer :=0;
22    signal alu_out: std_logic_vector(127 downto 0);
23
24    constant period : time := 20 ns;
25
26
27    ------------------------------------------------------------------------
    --
28    -- Instantiate and Map UUT
29    ------------------------------------------------------------------------
    --
30    file file input : text;
31    file file_output: text;
32
33
34
35    begin
36        UUT: entity pipiline_unit port map(
37            ins_in => rom_data,
38            clk => tb_clk,
39            ALU_o => tb_alu_out
40            );
41
42
43
44
    ------------------------------------------------------------------------
45    -- This procedure reads the file input_vectors.txt which is located in
    the
46    -- simulation project area.
47    -- It will read the data in and send it to the ripple-adder component
48    -- to perform the operations.  The result is written to the
49    -- output_results.txt file, located in the same directory.
50
    ------------------------------------------------------------------------
51    process
52      variable ILINE : line;
53      variable INS_LINE  : std_logic_vector(24 downto 0);
54      variable count: integer:=0;
```

```
55      variable v_out : line;
56
57    begin
58      file_open(file_input, "opcode.txt",  read_mode);
59      file_open(file_output, "alu_out.txt", write_mode);
60
61
62      while not endfile(file_input) loop
63        readline(file input, ILINE);
64        read(ILINE, INS_LINE);
65        rom_data(count)<= INS_LINE;
66        count := count+1;
67      end loop;
68      file_close(file_input);
69
70
71      for i in 0 to count +4 loop
72          wait for period;
73          write(v_out, tb_alu_out);
74          writeline(file_output, v_out);
75      end loop;
76      file_close(file_output);
77
78    end process;
79
80
81
82    clock: process
83    begin
84        tb_clk <='0';
85        wait for 10ns;
86        tb_clk<='1';
87        wait for 10ns;
88    end process;
89  end behave;
```