```vhdl
library IEEE;
use ieee.std_logic_1164.all;

library work;

entity andGate is
   port(
      A  :in std_logic;
      B     : in std_logic_vector(31 downto 0);
      S        : out std_logic_vector(31 downto 0)
   );
end andGate;

architecture behaviour of andGate is

begin
   process(A, B)
   begin
      for I in 0 to 31 loop
         S(I) <= A and B(I);
      end loop;
   end process;
end behaviour;
```

```vhdl
library IEEE;
use ieee.std_logic_1164.all;

library work;
entity andGate_32vs1 is
   port(
      A     : in std_logic;
      B     : in std_logic_vector(31 downto 0);
      S     : out std_logic_vector(31 downto 0)
   );
end andGate_32vs1;

architecture behaviour of andGate_32vs1 is

begin
   process(A, B)
   begin
      for I in 0 to 31 loop
         S(I) <= A and B(I);
      end loop;
   end process;
end behaviour;
```

```
1   library IEEE;
2   use ieee.std_logic_1164.all;
3
4   entity band5exp1 is
5    Port(
6      input1, input2, input3, input4, input5          : in std_logic;
7      output       : out std_logic);
8   end band5exp1;
9
10  architecture behaviour of band5exp1 is
11  begin
12
13   output <= (not input1) AND (not input2) AND (not input3) AND input4 and (not input5);
14
15  end behaviour;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity conFF is
   port(
      clk                              : in std_logic;
      IRout                            : in std_logic_vector(31 downto 0);
      BusMuxOut                        : in std_logic_vector(31 downto 0);
      CONout                           : out std_logic
   );
end entity;

architecture behaviour of conFF is

component decoder4bits is
   port (
      input  : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
      output : OUT STD_LOGIC_VECTOR(3 downto 0)
   );
end component;

component conFFSubComponent1 is
   port(
      decoderOutput                    : in std_logic_vector(3 downto 0);
      BusMuxOut                        : in std_logic_vector(31 downto 0);
      conFFOutput                      : out std_logic;
      busOrOut                         : inout std_logic
   );
end component;

component flipFlop is
   port(
      clk         : in std_logic;
      D           : in std_logic;
      Q           : out std_logic
   );
end component;

signal decoderInput  : std_logic_vector(1 downto 0);
signal decoderOutput : std_logic_vector(3 downto 0);
signal orGateToFlipFlop, conFFOutput, busOrOutInternal   : std_logic;

begin
decoderInput <= IRout(20 downto 19);

U0: decoder4bits port map(
      input  => decoderInput,
      output => decoderOutput
);

U1: conFFSubComponent1 port map(
      decoderOutput => decoderOutput,
      BusMuxOut => BusMuxOut,
      conFFOutput => orGateToFlipFlop,
      busOrOut => busOrOutInternal
);

U2: flipFlop port map(
      clk => clk,
      D => orGateToFlipFlop,
      Q => CONout
);
```

```
63    end architecture;
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity conFFSubComponent1 is
5      port(
6         decoderOutput                    : in std_logic_vector(3 downto 0);
7         BusMuxOut                        : in std_logic_vector(31 downto 0);
8         conFFOutput                      : out std_logic;
9         busOrOut : inout std_logic
10     );
11  end entity;
12
13  architecture behaviour of conFFSubComponent1 is
14  signal and0, and1, and2, and3, finalOrOutput: std_logic;
15  begin
16  process(BusMuxOut)
17  begin
18     busOrOut <= '0';
19     for I in 0 to 31 loop
20        if(BusMuxOut(I) = '1') then
21        busOrOut <= '1';
22        exit;
23        end if;
24     end loop;
25  end process;
26
27  and0 <= decoderOutput(0) and not busOrOut;
28  and1 <= decoderOutput(1) and busOrOut;
29  and2 <= decoderOutput(2) and not BusMuxOut(31);
30  and3 <= decoderOutput(3) and BusMuxOut(31);
31  finalOrOutput <= and0 or and1 or and2 or and3;
32  conFFOutput <= finalOrOutput;
33
34
35  end architecture;
36
```

```
-- Datapath Testbench
-- Date: 2018-04-05
-- Created by: Jonathan Saunders & Drew Harshaw

LIBRARY ieee;
LIBRARY std;
LIBRARY work;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_textio.all;
USE ieee.std_logic_unsigned.all;
USE std.textio.all;


ENTITY datapath_tb IS
END;

ARCHITECTURE datapath_tb_arc OF datapath_tb IS
   TYPE State IS(default, T0, T1, T2, T3, T4, T5, T6, T7, T8);
   TYPE opCode IS (ld, ldi, st, addi, andi, ori, br, jr, jal, mul, mfHI, mfLO, inIO, outIO);
   SIGNAL Present_state:   State := default;
   SIGNAL Present_instruction: opCode := ld;
   SIGNAL Clock_tb            : STD_LOGIC;
   SIGNAL busLine_tb          : STD_LOGIC_VECTOR(31 downto 0);
   SIGNAL read_notWrite_tb    :  STD_LOGIC;
   signal Rin_from_control_tb : STD_LOGIC;
   SIGNAL Rout_from_control_tb: STD_LOGIC;
   SIGNAL ram_done_cs_tb      : STD_LOGIC;
   SIGNAL clr_tb              :  STD_LOGIC;
   SIGNAL IOout_tb            : STD_LOGIC;
   SIGNAL Cout_tb             : STD_LOGIC;
   SIGNAL InPortout_tb        : STD_LOGIC;
   SIGNAL MDRout_tb           : STD_LOGIC;
   SIGNAL PCout_tb            : STD_LOGIC;
   SIGNAL Zlowout_tb          : STD_LOGIC;
   SIGNAL Zhighout_tb         : STD_LOGIC;
   SIGNAL LOout_tb            : STD_LOGIC;
   SIGNAL HIout_tb            : STD_LOGIC;
   SIGNAL regOut_tb           : std_logic_vector(15 downto 0);
   SIGNAL IOin_tb             :  STD_LOGIC;
   SIGNAL IO_to_inPort_tb     :  std_logic_vector (31 downto 0);
   SIGNAL outPort_to_IO_tb    :  std_logic_vector (31 downto 0);
   SIGNAL MDRin_tb            : STD_LOGIC;
   SIGNAL PCin_tb             : STD_LOGIC;
   SIGNAL IRin_tb             : STD_LOGIC;
   SIGNAL LOin_tb             : STD_LOGIC;
   SIGNAL HIin_tb             : STD_LOGIC;
   SIGNAL MARin_tb            : STD_LOGIC;
   SIGNAL Yin_tb              : STD_LOGIC;
   SIGNAL Zin_tb              : STD_LOGIC;
   signal alu : std_logic_vector(13 downto 0);
   SIGNAL add_cs_tb           : STD_LOGIC;
   signal sub_cs_tb           : STD_LOGIC;
   SIGNAL mult_cs_tb          : STD_LOGIC;
   SIGNAL div_cs_tb           : STD_LOGIC;
   SIGNAL shift_left_cs_tb    :  STD_LOGIC;
   SIGNAL shift_right_logical_cs_tb     : STD_LOGIC;
   SIGNAL shift_right_arithmetic_cs_tb  : STD_LOGIC;
   SIGNAL rotate_left_cs_tb   : STD_LOGIC;
   SIGNAL rotate_right_cs_tb  : STD_LOGIC;
   SIGNAL and_cs_tb           : STD_LOGIC;
   SIGNAL or_cs_tb            : STD_LOGIC;
   SIGNAL negate_cs_tb        : STD_LOGIC;
```

```
63      SIGNAL IncPC_tb            : STD_LOGIC;
64      SIGNAL not_cs_tb           : STD_LOGIC;
65      SIGNAL BAout_tb            : STD_LOGIC;
66      SIGNAL selGra_tb           : STD_LOGIC;
67      SIGNAL selGrb_tb           : STD_LOGIC;
68      SIGNAL selGrc_tb           : STD_LOGIC;
69      SIGNAL selRin_tb           : STD_LOGIC;
70      signal selRout_tb          : STD_LOGIC;
71      SIGNAL CON_in_tb : STD_LOGIC;
72      SIGNAL CON_to_control_tb   : STD_LOGIC;
73      signal readChannel_tb      : STD_LOGIC;
74
75
76  COMPONENT datapath is
77      PORT(
78          Clock, clr                  : in std_logic;
79          IO_to_inPort : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
80          MARin, MDRin, IOin, IOout, Zin, Yin, PCin, IRin, HIin, LOin :  IN std_logic;
81          Cout, InPortout, MDRout, PCout, Zlowout, Zhighout, Loout, HIout: in std_logic;
82          regOut      : Inout std_logic_vector(15 downto 0);
83          CON_in      :     in std_logic;
84          readWrite_to_memory, CON_to_control     :  OUT std_LOGIC;
85          shiftValue_to_control       :  OUT STD_LOGIC_VECTOR(4 downto 0);
86          BusMuxOut : INOUT std_LOGIC_VECTOR(31 downto 0);
87          ram_done_cs : inout std_logic;
88          read_notWrite      :     IN std_logic;
89          logicALUSelect  : in std_logic_vector(13 downto 0);
90          outPort_to_IO : out std_logic_vector(31 downto 0);
91
92          selGra, selGrb, selGrc, selRin, selRout, selBAout  :  IN std_logic
93
94      );
95  END COMPONENT datapath;
96  BEGIN
97
98
99  alu <= not_cs_tb & IncPC_tb & negate_cs_tb & OR_cs_tb & and_cs_tb & rotate_left_cs_tb &
    rotate_right_cs_tb & shift_right_arithmetic_cs_tb
100         & shift_right_logical_cs_tb & shift_left_cs_tb & div_cs_tb & mult_cs_tb & sub_cs_tb &
    ADD_cs_tb;
101
102
103     DUT0 : datapath   PORT MAP (
104         clock   => clock_tb,
105         clr   => clr_tb,
106         BusMuxOut   => busLine_tb,
107         IO_to_inPort   => IO_to_inPort_tb,
108         MARin   => MARin_tb,
109         MDRin   => MDRin_tb,
110         regOut => regOut_tb,
111         IOin   => IOin_tb,
112         IOout   => IOout_tb,
113         Zin   => Zin_tb,
114         Yin   => Yin_tb,
115         PCin   => PCin_tb,
116         IRin   => IRin_tb,
117         HIin   => HIin_tb,
118         LOin   => LOin_tb,
119         Cout  => Cout_tb,
120         InPortout => InPortout_tb,
121         MDRout => MDRout_tb,
122         PCout => PCout_tb,
```

```
123          Zlowout => Zlowout_tb,
124          Zhighout   => Zhighout_tb,
125          Loout  => LOout_tb ,
126          HIout => HIout_tb,
127          selGra   => selGra_tb,
128          selGrb   => selGrb_tb,
129          selGrc   => selGrc_tb,
130          selRin   => Rin_from_control_tb,
131          selRout   => Rout_from_control_tb,
132          selBAout   => BAout_tb,
133          CON_in => CON_in_tb,
134          ram_done_cs => ram_done_cs_tb,
135          read_notWrite   => read_notWrite_tb,
136          logicALUSelect => alu,
137          outPort_to_IO =>outPort_to_IO_tb,
138          CON_to_control   => CON_to_control_tb
139       );
140
141       Clock_process: PROCESS is
142       BEGIN
143          Clock_tb <= '1', '0' after 10 ns;
144          Wait for 20 ns;
145       END PROCESS Clock_process;
146
147       PROCESS(Clock_tb)
148       VARIABLE counter : integer range 0 to 15 := 0;
149       BEGIN
150          IF(rising_edge(Clock_tb))THEN
151             CASE Present_state IS
152                WHEN Default   =>
153                   counter := counter + 1;
154                   if(counter = 2)then
155                      counter := 0;
156                      Present_state <= T0;
157                      Present_instruction <= ldi;
158                   END IF;
159                WHEN T0 =>
160                   IF (ram_done_cs_tb = '1') then
161                      Present_state <= T1;
162                   end if;
163                WHEN T1 =>
164                   IF (ram_done_cs_tb = '1') then
165                         Present_state <= T2;
166                   end if;
167                WHEN T2 =>
168                   IF (ram_done_cs_tb = '1') then
169                      Present_state <= T3;
170                   end if;
171                WHEN T3 =>
172                   IF (ram_done_cs_tb = '1') then
173                      Present_state <= T4;
174                   end if;
175                WHEN T4 =>
176                   IF (ram_done_cs_tb = '1') then
177                      Present_state <= T5;
178                   end if;
179                WHEN T5 =>
180                   IF (ram_done_cs_tb = '1') then
181                      Present_state <= T6;
182                   end if;
183                WHEN T6 =>
184                   IF (ram_done_cs_tb = '1') then
```

```
185                     Present_state <= T7;
186                  end if;
187               WHEN T7 =>
188                  IF (ram_done_cs_tb = '1') then
189                     Present_state <= T8;
190                  end if;
191               WHEN T8 =>
192                  IF (ram_done_cs_tb = '1') then
193                     Present_state <= T0;
194                     CASE Present_instruction IS
195                        WHEN  ldi =>
196                           Present_instruction <= ld;
197                        WHEN  ld =>
198                           Present_instruction <= addi;
199                        WHEN  addi =>
200                           Present_instruction <= ori;
201                        WHEN  ori =>
202                           Present_instruction <= andi;
203                        WHEN  andi =>
204                           Present_instruction <= br;
205                        WHEN  br =>
206                           counter := counter + 1;
207                           if (counter = 4) then
208                              Present_instruction <= jr;
209                              counter := 0;
210                           end if;
211                        WHEN jr =>
212                           Present_instruction <= jal;
213                        WHEN jal =>
214                           Present_instruction <= mul;
215                        WHEN mul =>
216                           Present_instruction <= mfHI;
217                        WHEN mfHI =>
218                           Present_instruction <= mfLO;
219                        WHEN mfLO =>
220                           Present_instruction <= outIO;
221                        WHEN outIO =>
222                           Present_instruction <= inIO;
223                        WHEN inIO =>
224                           Present_instruction <= st;
225                        WHEN st =>
226                           counter := counter + 1;
227                           if (counter = 9) then
228                              Present_instruction <= ldi;
229                              counter := 0;
230                           end if;
231                        WHEN others =>
232                           Present_instruction <= ldi;
233                        END CASE;
234                  end if;
235               WHEN OTHERS =>
236                  Present_state <= Default;
237            END CASE;
238         END IF;
239      END PROCESS;
240
241      PROCESS (Present_state)
242      BEGIN
243         CASE Present_state IS
244      WHEN Default =>
245
246         MARin_tb <= '0'; Zin_tb <= '0';
```

```
247         PCin_tb <= '0'; MDRin_tb <= '0'; IRin_tb <= '0'; Yin_tb <= '0';
248         IncPC_tb <= '0'; read_notWrite_tb <= '0', '1' after 1 ns; AND_cs_tb <= '0';
249         div_cs_tb <= '0'; HIin_tb <= '0'; IOout_tb <= '0'; LOout_tb <= '0'; HIout_tb <= '0';
    Zhighout_tb <= '0';
250         shift_left_cs_tb <= '0'; rotate_left_cs_tb <= '0'; mult_cs_tb <= '0';
    rotate_right_cs_tb <= '0'; shift_right_arithmetic_cs_tb <= '0';
251         negate_cs_tb <= '0'; shift_right_logical_cs_tb <= '0'; LOin_tb <= '0'; Cout_tb <= '0';
     ADD_cs_tb <= '0'; not_cs_tb <= '0';
252         or_cs_tb <= '0'; sub_cs_tb <= '0'; IOin_tb <= '0'; InPortout_tb <= '0';
253
254         PCout_tb <= '0';Zlowout_tb <= '0'; MDRout_tb <= '0';
255         Zhighout_tb <= '0'; Zlowout_tb <= '0';
256         HIout_tb <= '0';
257         selRin_tb <= '0'; selRout_tb <= '0';
258         selGra_tb <= '0'; selGrb_tb <= '0'; selGrc_tb <= '0'; Rin_from_control_tb <= '0';
    Rout_from_control_tb <= '0'; BAout_tb <= '0';
259         CON_in_tb <= '0';
260         IO_to_inPort_tb <= x"0000ffff";
261
262         --start system
263         clr_tb <= '0', '1' after 15 ns;
264
265      WHEN T0 =>
266               --turn on signals
267               PCout_tb <= '1';
268               MARin_tb <= '1';
269               Zin_tb   <= '1';
270               IncPC_tb <= '1';
271      WHEN T1 =>
272               --turn off signals
273               PCout_tb <= '0';
274               MARin_tb <= '0';
275               Zin_tb   <= '0';
276               IncPC_tb <= '0';
277
278               --turn on signals
279               Zlowout_tb  <= '1';
280               PCin_tb     <= '1';
281               read_notWrite_tb  <= '1';
282               MDRin_tb    <= '1';
283      WHEN T2 =>
284               --turn off signals
285               Zlowout_tb  <= '0';
286               PCin_tb     <= '0';
287               MDRin_tb    <= '0';
288
289               --turn on signals
290               MDRout_tb   <= '1';
291               IRin_tb  <= '1';
292      WHEN T3 =>
293               --turn off signals
294               MDRout_tb   <= '0';
295               IRin_tb  <= '0';
296
297               --turn on signals
298               CASE Present_instruction IS
299                     WHEN ld =>
300                        selGrb_tb <= '1'; BAout_tb <= '1'; Yin_tb <= '1'; selRin_tb <= '1';
301                     WHEN ldi =>
302                        selGrb_tb <= '1'; BAout_tb <= '1'; Yin_tb <= '1'; selRin_tb <= '1';
303                     WHEN st =>
304                        selGrb_tb <= '1'; BAout_tb <= '1'; Yin_tb <= '1';
```

```
305                     WHEN addi | andi | ori | mul =>
306                        selGrb_tb <= '1'; Rout_from_control_tb <= '1'; Yin_tb <= '1';
307                     WHEN br =>
308                        selGra_tb <= '1'; Rout_from_control_tb <= '1'; CON_in_tb <= '1' after 3
       ns;
309                     WHEN jal =>
310                        PCout_tb <= '1'; Rin_from_control_tb <= '1'; selGrb_tb <= '1';
311                     WHEN jr =>
312                        selGra_tb <= '1'; Rout_from_control_tb <= '1'; PCin_tb <= '1';
313                     WHEN mfHI =>
314                        HIout_tb <= '1'; Rin_from_control_tb <= '1'; selGra_tb <= '1';
315                     WHEN mfLO =>
316                        LOout_tb <= '1'; Rin_from_control_tb <= '1'; selGra_tb <= '1';
317                     WHEN outIO =>
318                        IOout_tb <= '1'; selGra_tb <= '1'; Rout_from_control_tb <= '1';
319                     WHEN inIO =>
320                        InPortout_tb <= '1'; selGra_tb <= '1'; Rin_from_control_tb <= '1';
321                     WHEN OTHERS =>
322                  END CASE;
323
324       WHEN T4 =>
325             CASE Present_instruction IS
326                  WHEN ld =>
327                     selGrb_tb <= '0'; BAout_tb <= '0'; Yin_tb <= '0'; selRin_tb <= '0';
328                     Cout_tb <= '1'; ADD_cs_tb <= '1'; Zin_tb <= '1';
329                  WHEN ldi =>
330                     selGrb_tb <= '0'; BAout_tb <= '0'; Yin_tb <= '0';selRin_tb <= '0';
331                     Cout_tb <= '1'; ADD_cs_tb <= '1'; Zin_tb <= '1';
332                  WHEN st =>
333                     selGrb_tb <= '0'; BAout_tb <= '0'; Yin_tb <= '0';
334                     Cout_tb <= '1'; ADD_cs_tb <= '1'; Zin_tb <= '1';
335                  WHEN addi =>
336                     selGrb_tb <= '0'; Rout_from_control_tb <= '0'; Yin_tb <= '0';
337                     Cout_tb <= '1'; ADD_cs_tb <= '1'; Zin_tb <= '1';
338                  WHEN ori =>
339                     selGrb_tb <= '0'; Rout_from_control_tb <= '0'; Yin_tb <= '0';
340                     Cout_tb <= '1'; OR_cs_tb <= '1'; Zin_tb <= '1';
341                  WHEN andi =>
342                     selGrb_tb <= '0'; Rout_from_control_tb <= '0'; Yin_tb <= '0';
343                     Cout_tb <= '1'; AND_cs_tb <= '1'; Zin_tb <= '1';
344                  WHEN mul =>
345                     selGrb_tb <= '0'; Yin_tb <= '0';
346                     selGra_tb <= '1'; mult_cs_tb <= '1'; Zin_tb <= '1';
347                  WHEN br =>
348                     selGra_tb <= '0'; Rout_from_control_tb <= '0'; CON_in_tb <= '0';
349                     PCout_tb <= '1'; Yin_tb <= '1';
350                  WHEN jal =>
351                     PCout_tb <= '0'; Rin_from_control_tb <= '0'; selGrb_tb <= '0';
352                     selGra_tb <= '1'; Rout_from_control_tb <= '1'; PCin_tb <= '1';
353                  WHEN jr =>
354                     selGra_tb <= '0'; Rout_from_control_tb <= '0'; PCin_tb <= '0';
355                  WHEN mfHI | mfLO =>
356                     HIout_tb <= '0'; LOout_tb <= '0'; Rin_from_control_tb <= '0'; selGra_tb <=
       '0';
357                  WHEN outIO | inIO =>
358                     InPortout_tb <= '0'; selGra_tb <= '0'; Rin_from_control_tb <= '0';
      IOout_tb <= '0'; Rout_from_control_tb <= '0';
359                  WHEN OTHERS =>
360               END CASE;
361       WHEN T5 =>
362                CASE Present_instruction IS
363                     WHEN ldi =>
```

```
364                      Cout_tb <= '0'; ADD_cs_tb <= '0'; Zin_tb <= '0';
365                      Zlowout_tb <= '1'; selGra_tb <= '1'; Rin_from_control_tb <= '1';
366                   when ld =>
367                      Cout_tb <= '0'; ADD_cs_tb <= '0'; Zin_tb <= '0';
368                      Zlowout_tb <= '1'; MARin_tb <= '1';
369                   WHEN st =>
370                      Cout_tb <= '0'; ADD_cs_tb <= '0'; Zin_tb <= '0';
371                      Zlowout_tb <= '1'; MARin_tb <= '1';
372                   WHEN addi | ori | andi =>
373                      Cout_tb <= '0'; ADD_cs_tb <= '0'; AND_cs_tb <= '0'; OR_cs_tb <= '0';
    Zin_tb <= '0';
374                      Zlowout_tb <= '1'; selGra_tb <= '1'; Rin_from_control_tb <= '1';
375                   WHEN mul =>
376                      selGra_tb <= '0'; mult_cs_tb <= '0'; Zin_tb <= '0';
    Rout_from_control_tb <= '0';
377                      Zlowout_tb <= '1'; LOin_tb <= '1';
378                   WHEN br =>
379                      PCout_tb <= '0'; Yin_tb <= '0';
380                      Cout_tb <= '1'; ADD_cs_tb <= '1'; Zin_tb <= '1';
381                   WHEN jal =>
382                      selGra_tb <= '0'; Rout_from_control_tb <= '0'; PCin_tb <= '0';
383                   WHEN inIO =>
384                      selGra_tb <= '0'; Rin_from_control_tb <= '0';
385                   WHEN OTHERS =>
386                END CASE;
387      WHEN T6 =>
388               CASE Present_instruction IS
389                   WHEN ld =>
390                      Zlowout_tb <= '0'; MARin_tb <= '0';
391                      read_notWrite_tb <= '1'; MDRin_tb <= '1';
392                   when ldi =>
393                      Zlowout_tb <= '0'; selGra_tb <= '0'; BAout_tb <= '0';
    Rin_from_control_tb <= '0';
394                   WHEN st =>
395                      Zlowout_tb <= '0'; MARin_tb<= '0';
396                      MDRin_tb <= '1'; selGra_tb <= '1'; BAout_tb <= '1'; read_notWrite_tb <=
     '0';
397                   WHEN addi | andi | ori =>
398                      Zlowout_tb <= '0'; selGra_tb <= '0'; Rin_from_control_tb <= '0';
399                   WHEN mul =>
400                      Zlowout_tb <= '0'; LOin_tb <= '0';
401                      Zhighout_tb <= '1'; HIin_tb <= '1';
402                   WHEN br =>
403                      Cout_tb <= '0'; ADD_cs_tb <= '0'; Zin_tb <= '0';
404                      Zlowout_tb <= '1';
405                      if (CON_to_control_tb = '1') THEN
406                         PCin_tb <= '1';
407                      end if;
408                   WHEN OTHERS =>
409               END CASE;
410      WHEN T7 =>
411               CASE Present_instruction IS
412                   when ldi =>
413                   WHEN ld =>
414                      MDRin_tb <= '0';
415                      MDRout_tb <= '1'; selGra_tb <= '1'; Rin_from_control_tb <= '1';
416                   WHEN st =>
417                      MDRin_tb <= '0'; selGra_tb <= '0'; BAout_tb <= '0';
418                   WHEN mul =>
419                      Zhighout_tb <= '0'; HIin_tb <= '0';
420                   WHEN br =>
421                      Zlowout_tb <= '0'; PCin_tb <= '0';
```

```
422                     WHEN OTHERS =>
423                END CASE;
424        WHEN T8 =>
425                CASE Present_instruction IS
426                     WHEN ld =>
427                        MDRout_tb <= '0'; selGra_tb <= '0'; Rin_from_control_tb <= '0';
428                     WHEN st =>
429                        read_notWrite_tb <= '1';
430                     WHEN OTHERS =>
431                END CASE;
432             WHEN OTHERS =>
433          END CASE;
434       END PROCESS;
435    END ARCHITECTURE datapath_tb_arc;
436
437
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity datapath is
5      port(
6         Clock, clr                    : in std_logic;
7         IO_to_inPort : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
8         MARin, MDRin, IOin, IOout, Zin, Yin, PCin, IRin, HIin, LOin :  IN std_logic;
9         Cout, InPortout, MDRout, PCout, Zlowout, Zhighout, Loout, HIout: in std_logic;
10        regOut       : Inout std_logic_vector(15 downto 0);
11        selGra, selGrb, selGrc, selRin, selRout, selBAout  :  IN std_logic;
12        CON_in       :     in std_logic;
13        ram_done_cs : inout std_logic;
14        read_notWrite      :     IN std_logic;
15        logicALUSelect  : in std_logic_vector(13 downto 0);
16        outPort_to_IO : out std_logic_vector(31 downto 0);
17        readWrite_to_memory, CON_to_control     :  OUT std_LOGIC;
18        shiftValue_to_control       :  OUT STD_LOGIC_VECTOR(4 downto 0);
19        BusMuxOut : INOUT std_LOGIC_VECTOR(31 downto 0);
20        memoryData_to_computerSystem   :  OUT std_LOGIC_VECTOR(31 downto 0)
21     );
22  end datapath;
23
24  architecture datapath_arc of datapath is
25
26  component ALU
27     port(
28        control                  : in std_logic_vector(13 downto 0);
29        A, B                     : in std_logic_vector(31 downto 0);
30        C                     : out std_logic_vector(63 downto 0)
31     );
32  end component;
33
34
35  component IO_Units
36     port(
37        clk :  IN  STD_LOGIC;
38        clr :  IN  STD_LOGIC;
39        In_cs :  IN  STD_LOGIC;
40        Out_cs :  IN  STD_LOGIC;
41        toInPort :  IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
42        toOutPort :  IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
43        toBus :  OUT  STD_LOGIC_VECTOR(31 DOWNTO 0);
44        toIO :  OUT  STD_LOGIC_VECTOR(31 DOWNTO 0)
45     );
46  end component;
47
48
49  component zRegister
50     port(
51        C : in std_logic_vector(63 downto 0);
52        Zin, clk : IN STD_LOGIC;
53        Zhigh, Zlow : out std_logic_vector(31 downto 0)
54     );
55  end component;
56
57
58  component registerFile
59     port
60     (
61        clk, clr, BAout    : in std_logic;
62        Rin    : in std_logic_vector(15 downto 0);
```

```
63          BusMuxOut    : in std_logic_vector(31 downto 0);
64          BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3,
65          BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
66          BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11,
67          BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15 : out std_logic_vector(31 downto 0)
68       );
69    end component;
70
71
72    component reg_32
73       port(
74          clk          : in std_logic;
75          clr          : in std_logic;
76          Rin          : in std_logic;
77          BusMuxOut    : in std_logic_vector(31 downto 0);
78          BusMuxIn     : out std_logic_vector(31 downto 0)
79       );
80    end component;
81
82
83    component multiplexer32bits
84       port (
85          BusMuxIn_R0, BusMuxIn_R1, BusMuxIn_R2, BusMuxIn_R3,
86          BusMuxIn_R4, BusMuxIn_R5, BusMuxIn_R6, BusMuxIn_R7,
87          BusMuxIn_R8, BusMuxIn_R9, BusMuxIn_R10, BusMuxIn_R11,
88          BusMuxIn_R12, BusMuxIn_R13, BusMuxIn_R14, BusMuxIn_R15,
89          BusMuxIn_HI, BusMuxIn_LO, BusMuxIn_Zhigh, BusMuxIn_Zlow,
90          BusMuxIn_PC, BusMuxIn_MDR, BusMuxIn_InPort,
91          C_sign_extended: IN STD_LOGIC_VECTOR(31 DOWNTO 0);
92
93          BusMuxOut       : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
94          encoderSignal   : IN STD_LOGIC_VECTOR(4 DOWNTO 0)
95       );
96    end component;
97
98    component multiplexerMDR
99       port(
100         BusMuxOut, Mdatain: IN STD_LOGIC_VECTOR(31 DOWNTO 0);
101         ReadChannel        : IN STD_LOGIC;
102         MDRMuxOut          : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
103         );
104   end component;
105
106   component encoder32bits
107      port (
108         input  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
109         output : OUT STD_LOGIC_VECTOR(4 downto 0)
110      );
111   end component;
112
113   COMPONENT conFF
114      PORT
115      (
116         clk      :   IN STD_LOGIC;
117         IRout    :   IN STD_LOGIC_VECTOR(31 DOWNTO 0);
118         BusMuxOut       :   IN STD_LOGIC_VECTOR(31 DOWNTO 0);
119         CONout      :   OUT STD_LOGIC
120      );
121   END COMPONENT;
122
123   component selectAndEncodeLogic is
124      port(
```

```
125         IRin                                : in std_logic_vector(31 downto 0);
126         Gra, Grb, Grc, Rin, Rout, BAout     : in std_logic;
127         BusMuxOut                           : in std_logic_vector(31 downto 0);
128         C_sign_extended                     : out std_logic_vector(31 downto 0);
129         r0in_r15in_Decoded                  : out std_logic_vector(15 downto 0);
130         r0out_r15out_Decoded                : out std_logic_vector(15 downto 0)
131      );
132   end component;
133
134   component memorySubsystem is
135      port(
136         BusMuxOut    : in std_logic_vector(31 downto 0);
137         BusMuxInMDR  : inout std_logic_vector(31 downto 0);
138         MDRin, MARin, clock, clear: in std_logic;
139         read_notWrite  : in std_logic;
140         ram_done_cs    : inout std_logic
141      );
142   end component;
143
144
145
146   signal   busR0, busR1, busR2, busR3,
147         busR4, busR5, busR6, busR7,
148         busR8, busR9, busR10, busR11,
149         busR12, busR13, busR14, busR15 : std_logic_vector(31 downto 0);
150
151   signal   busPCin, busIRin, busMARin, busMDRin,
152            busInPortin, busOutPortin, busHIin, busLOin,
153            busZhighin, busZlowin, busSignExtendedIn   : std_logic_vector(31 downto 0);
154   signal registerFileIn : std_logic_vector(15 downto 0);
155
156
157   signal encoderControlBus : std_logic_vector(4 downto 0);
158
159
160   signal YtoA : std_logic_vector(31 downto 0);
161   signal CtoZ : std_logic_vector(63 downto 0);
162
163   signal complete : std_logic;
164
165   begin
166
167   PC: reg_32  port map(
168         clk => Clock,
169         clr   => clr,
170         Rin => PCin,
171         BusMuxOut => BusMuxOut,
172         BusMuxIn => busPCin
173      );
174
175   IR: reg_32  port map(
176         clk => Clock,
177         clr   => clr,
178         Rin => IRin,
179         BusMuxOut => BusMuxOut,
180         BusMuxIn => busIRin
181      );
182
183   IO: IO_Units port map(
184         clk => Clock,
185         clr => clr,
186         In_cs => IOin,
```

```
187         Out_cs => IOout,
188         toInPort => IO_to_inPort,
189         toOutPort => BusMuxOut,
190         toBus => busInPortin,
191         toIO => outPort_to_IO
192       );
193
194
195   HI: reg_32  port map(
196         clk => Clock,
197         clr   => clr,
198         Rin => HIin,
199         BusMuxOut => BusMuxOut,
200         BusMuxIn => busHIin
201   );
202
203   LO: reg_32  port map(
204         clk => Clock,
205         clr   => clr,
206         Rin => LOin,
207         BusMuxOut => BusMuxOut,
208         BusMuxIn => busLOin
209   );
210
211   datapath_register_file : registerFile port map(
212         clk => Clock,
213         clr => clr,
214         BAout => selBAout,
215         Rin => registerFileIn,
216         BusMuxOut => BusMuxOut,
217         BusMuxInR0 => busR0,
218         BusMuxInR1 => busR1,
219         BusMuxInR2 => busR2,
220         BusMuxInR3 => busR3,
221         BusMuxInR4 => busR4,
222         BusMuxInR5 => busR5,
223         BusMuxInR6 => busR6,
224         BusMuxInR7 => busR7,
225         BusMuxInR8 => busR8,
226         BusMuxInR9 => busR9,
227         BusMuxInR10 => busR10,
228         BusMuxInR11 => busR11,
229         BusMuxInR12 => busR12,
230         BusMuxInR13 => busR13,
231         BusMuxInR14 => busR14,
232         BusMuxInR15 => busR15
233       );
234
235
236   Y: reg_32   port map(
237         clk => Clock,
238         clr   => clr,
239         Rin => Yin,
240         BusMuxOut => BusMuxOut,
241         BusMuxIn => YtoA
242   );
243
244   Z: zRegister port map(
245         C => CtoZ,
246         clk => Clock,
247         Zin => Zin,
248         Zhigh => busZhighin,
```

```
249         Zlow => busZlowin
250     );
251
252
253   Encoder: encoder32bits port map(
254         input(31 downto 24)  => x"00",
255         input(23)    => Cout,
256         input(22)    => InPortout,
257         input(21)    => MDRout,
258         input(20)    => PCout,
259         input(19)    => Zlowout,
260         input(18)    => Zhighout,
261         input(17)    => Loout,
262         input(16)    => HIout,
263         input(15 downto 0)    => regOut,
264         output    => encoderControlBus
265     );
266
267   Multiplexer: multiplexer32bits port map(
268         BusMuxIn_R0     => busR0,
269         BusMuxIn_R1     => busR1,
270         BusMuxIn_R2     => busR2,
271         BusMuxIn_R3     => busR3,
272         BusMuxIn_R4     => busR4,
273         BusMuxIn_R5     => busR5,
274         BusMuxIn_R6     => busR6,
275         BusMuxIn_R7     => busR7,
276         BusMuxIn_R8     => busR8,
277         BusMuxIn_R9     => busR9,
278         BusMuxIn_R10    => busR10,
279         BusMuxIn_R11    => busR11,
280         BusMuxIn_R12    => busR12,
281         BusMuxIn_R13    => busR13,
282         BusMuxIn_R14    => busR14,
283         BusMuxIn_R15    => busR15,
284         BusMuxIn_HI     => busHIin,
285         BusMuxIn_LO     => busLOin,
286         BusMuxIn_Zhigh  => busZhighin,
287         BusMuxIn_Zlow   => busZlowin,
288         BusMuxIn_PC     => busPCin,
289         BusMuxIn_MDR    => BusMDRIn,
290         BusMuxIn_InPort => busInPortin,
291         C_sign_extended => busSignExtendedIn,
292         BusMuxOut       => BusMuxOut,
293         encoderSignal   => encoderControlBus
294     );
295
296   Arithmetic: ALU port map(
297         control => logicALUSelect,
298         A => YtoA,
299         B => BusMuxOut,
300         C => CtoZ
301     );
302
303   SelectEncode: selectAndEncodeLogic port map(
304         IRin=> busIRin,
305         Gra => selGra,
306         Grb => selGrb,
307         Grc => selGrc,
308         Rin => selRin,
309         Rout => selRout,
310         BAout => selBAout,
```

```
311         BusMuxOut => BusMuxOut,
312         C_sign_extended => busSignExtendedIn,
313         r0in_r15in_Decoded => registerFileIn(15 downto 0),
314         r0out_r15out_Decoded => regOut
315     );
316
317     Memory: memorySubsystem port map(
318         BusMuxOut => BusMuxOut,
319         ram_done_cs => ram_done_cs,
320         BusMuxInMDR => BusMDRIn,
321         MDRin => MDRin,
322         MARin => MARin,
323         clock => Clock,
324         clear => clr,
325         read_notWrite => read_notWrite
326     );
327
328     conFFLogic: conFF PORT MAP(
329         clk   => CON_in,
330         IRout => busIRin,
331         BusMuxOut => BusMuxOut,
332         CONout   => con_to_control
333     );
334     end architecture datapath_arc;
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY decoder4bits IS
5      PORT (
6
7         input  : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
8         output : OUT STD_LOGIC_VECTOR(3 downto 0)
9
10     );
11  end entity decoder4bits;
12
13  architecture behavioural of decoder4bits is
14  begin
15
16     output<="0001" when input = "00"
17        else "0010" when input = "01"
18        else "0100" when input = "10"
19        else "1000" when input = "11";
20
21
22  end architecture behavioural;
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY encoder32bits IS
5      PORT (
6
7         input  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
8         output : OUT STD_LOGIC_VECTOR(4 downto 0)
9
10     );
11  end entity encoder32bits;
12
13  architecture behavioural of encoder32bits is
14  begin
15
16     output<="00000" when input = x"00000000"
17        else "00001" when input = x"00000001"
18        else "00010" when input = x"00000002"
19        else "00011" when input = x"00000004"
20        else "00100" when input = x"00000008"
21        else "00101" when input = x"00000010"
22        else "00110" when input = x"00000020"
23        else "00111" when input = x"00000040"
24        else "01000" when input = x"00000080"
25        else "01001" when input = x"00000100"
26        else "01010" when input = x"00000200"
27        else "01011" when input = x"00000400"
28        else "01100" when input = x"00000800"
29        else "01101" when input = x"00001000"
30        else "01110" when input = x"00002000"
31        else "01111" when input = x"00004000"
32        else "10000" when input = x"00008000"
33        else "10001" when input = x"00010000"
34        else "10010" when input = x"00020000"
35        else "10011" when input = x"00040000"
36        else "10100" when input = x"00080000"
37        else "10101" when input = x"00100000"
38        else "10110" when input = x"00200000"
39        else "10111" when input = x"00400000";
40
41  end architecture behavioural;
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   entity flipFlop is
5      port(
6         clk         : in std_logic;
7         D           : in std_logic;
8         Q           : out std_logic
9      );
10  end entity flipFlop;
11
12  architecture behaviour of flipFlop is
13  begin
14     process (clk) is
15     begin
16        if rising_edge(clk) then
17              Q <= D;
18        end if;
19     end process;
20  end architecture behaviour;
```

```
1   -- megafunction wizard: %LPM_ADD_SUB%
2   -- GENERATION: STANDARD
3   -- VERSION: WM1.0
4   -- MODULE: LPM_ADD_SUB
5
6   -- ============================================================
7   -- File Name: increment.vhd
8   -- Megafunction Name(s):
9   --          LPM_ADD_SUB
10  --
11  -- Simulation Library Files(s):
12  --          lpm
13  -- ============================================================
14  -- ************************************************************
15  -- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
16  --
17  -- 13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
18  -- ************************************************************
19
20
21  --Copyright (C) 1991-2013 Altera Corporation
22  --Your use of Altera Corporation's design tools, logic functions
23  --and other software and tools, and its AMPP partner logic
24  --functions, and any output files from any of the foregoing
25  --(including device programming or simulation files), and any
26  --associated documentation or information are expressly subject
27  --to the terms and conditions of the Altera Program License
28  --Subscription Agreement, Altera MegaCore Function License
29  --Agreement, or other applicable license agreement, including,
30  --without limitation, that your use is for the sole purpose of
31  --programming logic devices manufactured by Altera and sold by
32  --Altera or its authorized distributors.  Please refer to the
33  --applicable agreement for further details.
34
35
36  LIBRARY ieee;
37  USE ieee.std_logic_1164.all;
38
39  LIBRARY lpm;
40  USE lpm.all;
41
42  ENTITY increment IS
43     PORT
44     (
45        datab     : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
46        result      : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
47     );
48  END increment;
49
50
51  ARCHITECTURE SYN OF increment IS
52
53     SIGNAL sub_wire0   : STD_LOGIC_VECTOR (31 DOWNTO 0);
54     SIGNAL sub_wire1_bv   : BIT_VECTOR (31 DOWNTO 0);
55     SIGNAL sub_wire1   : STD_LOGIC_VECTOR (31 DOWNTO 0);
56
57
58
59     COMPONENT lpm_add_sub
60     GENERIC (
61        lpm_direction     : STRING;
62        lpm_hint    : STRING;
```

```
63          lpm_representation      : STRING;
64          lpm_type    : STRING;
65          lpm_width       : NATURAL
66       );
67       PORT (
68             dataa : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
69             datab : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
70             result   : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
71       );
72       END COMPONENT;
73
74    BEGIN
75       sub_wire1_bv(31 DOWNTO 0) <= "00000000000000000000000000000001" ;
76       sub_wire1    <= To_stdlogicvector(sub_wire1_bv);
77       result    <= sub_wire0(31 DOWNTO 0);
78
79       LPM_ADD_SUB_component : LPM_ADD_SUB
80       GENERIC MAP (
81          lpm_direction => "ADD",
82          lpm_hint => "ONE_INPUT_IS_CONSTANT=YES,CIN_USED=NO" ,
83          lpm_representation => "UNSIGNED",
84          lpm_type => "LPM_ADD_SUB",
85          lpm_width => 32
86       )
87       PORT MAP (
88          dataa => sub_wire1,
89          datab => datab,
90          result => sub_wire0
91       );
92
93
94
95    END SYN;
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3
4   ENTITY IO_Units IS
5      PORT
6      (
7         clk :  IN  STD_LOGIC;
8         clr :  IN  STD_LOGIC;
9         In_cs :  IN  STD_LOGIC;
10        Out_cs :  IN  STD_LOGIC;
11        toInPort :  IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
12        toOutPort :  IN  STD_LOGIC_VECTOR(31 DOWNTO 0);
13        toBus :  OUT  STD_LOGIC_VECTOR(31 DOWNTO 0);
14        toIO :  OUT  STD_LOGIC_VECTOR(31 DOWNTO 0)
15     );
16  END IO_Units;
17
18  ARCHITECTURE behaviour OF IO_Units IS
19
20  COMPONENT reg_32
21     PORT(
22        clk, clr, Rin : IN STD_LOGIC;
23        BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
24        BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
25        );
26  END COMPONENT;
27
28  BEGIN
29
30  InPort : reg_32 PORT MAP(
31         clk => clk,
32         clr => clr,
33         Rin => In_cs,
34         BusMuxOut => toInPort,
35         BusMuxIn => toBus);
36
37
38  OutPort : reg_32 PORT MAP(
39         clk => clk,
40         clr => clr,
41         Rin => Out_cs,
42         BusMuxOut => toOutPort,
43         BusMuxIn => toIO);
44
45
46  END behaviour;
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity memorySubsystem is
5     port(
6        BusMuxOut    : in std_logic_vector(31 downto 0);
7        ram_done_cs  : inout std_logic;
8        BusMuxInMDR : inout std_logic_vector(31 downto 0);
9        MDRin, MARin, clock, clear: in std_logic;
10       read_notWrite : in std_logic
11    );
12 end entity;
13
14 architecture behaviour of memorySubsystem is
15
16 component multiplexerMDR is
17    port(
18       BusMuxOut, Mdatain: IN STD_LOGIC_VECTOR(31 DOWNTO 0);
19       ReadChannel : IN STD_LOGIC;
20       MDRMuxOut : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
21    );
22 end component;
23
24 component reg_32 is
25    port(
26       clk, clr, Rin : IN STD_LOGIC;
27       BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
28       BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
29    );
30 end component;
31
32 component regMAR IS
33    PORT(
34       clk, clr, Rin : IN STD_LOGIC;
35       BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
36       BusMuxIn : OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
37       );
38 END component;
39
40
41 component ram IS
42    PORT(
43       address     : IN STD_LOGIC_VECTOR (8 DOWNTO 0);
44       clock    : IN STD_LOGIC  := '1';
45       data     : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
46       rden     : IN STD_LOGIC  := '1';
47       wren     : IN STD_LOGIC ;
48       q     : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
49    );
50 END component;
51
52 signal mdMuxToMDR, mdrToRam, BusMuxInRAM : std_logic_vector(31 downto 0);
53 signal address : std_logic_vector(8 downto 0);
54 signal writeSig      : std_logic;
55 signal complete : std_logic;
56 begin
57
58
59 writeSig <= not read_notWrite;
60
61 ramCompletionSignalGeneration  : process (BusMuxInMDR(8 DOWNTO 0), writeSig, address)
62 begin
```

```
63      complete <= '0', '1' after 45 ns;
64  end process;
65  ram_done_cs <= complete;
66
67
68
69  --MDMUX
70  U0: multiplexerMDR port map(
71        BusMuxOut => BusMuxOut,
72        Mdatain => BusMuxInRAM,
73        ReadChannel => read_notWrite,
74        MDRMuxOut => mdMuxToMDR
75  );
76  --MDR Reg
77  U1: reg_32 port map(
78     clk => clock,
79     clr => clear,
80     Rin => MDRin,
81     BusMuxOut => mdMuxToMDR,
82     BusMuxIn => BusMuxInMDR
83  );
84  --MAR
85  U2: regMAR port map(
86     clk => clock,
87     clr => clear,
88     Rin => MARin,
89     BusMuxOut => BusMuxOut,
90     BusMuxIn => address
91  );
92  --RAM
93  U3: ram port map(
94        address => address,
95        clock => clock,
96        data => BusMuxInMDR,
97        rden => read_notWrite,
98        wren => writeSig,
99        q => BusMuxInRAM
100 );
101 end architecture;
102
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY multiplexer32bits IS
5      PORT (
6         BusMuxIn_R0, BusMuxIn_R1, BusMuxIn_R2, BusMuxIn_R3,
7         BusMuxIn_R4, BusMuxIn_R5, BusMuxIn_R6, BusMuxIn_R7,
8         BusMuxIn_R8, BusMuxIn_R9, BusMuxIn_R10, BusMuxIn_R11,
9         BusMuxIn_R12, BusMuxIn_R13, BusMuxIn_R14, BusMuxIn_R15,
10        BusMuxIn_HI, BusMuxIn_LO, BusMuxIn_Zhigh, BusMuxIn_Zlow,
11        BusMuxIn_PC, BusMuxIn_MDR, BusMuxIn_InPort,
12        C_sign_extended: IN STD_LOGIC_VECTOR(31 DOWNTO 0);
13        encoderSignal : IN STD_LOGIC_VECTOR(4 DOWNTO 0) ;
14        BusMuxOut : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
15     );
16  end entity multiplexer32bits;
17
18  architecture behavioural of multiplexer32bits is
19  begin
20     process (encoderSignal)
21     begin
22     case encoderSignal is
23        when "00000" => BusMuxOut <= BusMuxIn_R0;
24        when "00001" => BusMuxOut <= BusMuxIn_R1;
25        when "00010" => BusMuxOut <= BusMuxIn_R2;
26        when "00011" => BusMuxOut <= BusMuxIn_R3;
27        when "00100" => BusMuxOut <= BusMuxIn_R4;
28        when "00101" => BusMuxOut <= BusMuxIn_R5;
29        when "00110" => BusMuxOut <= BusMuxIn_R6;
30        when "00111" => BusMuxOut <= BusMuxIn_R7;
31        when "01000" => BusMuxOut <= BusMuxIn_R8;
32        when "01001" => BusMuxOut <= BusMuxIn_R9;
33        when "01010" => BusMuxOut <= BusMuxIn_R10;
34        when "01011" => BusMuxOut <= BusMuxIn_R11;
35        when "01100" => BusMuxOut <= BusMuxIn_R12;
36        when "01101" => BusMuxOut <= BusMuxIn_R13;
37        when "01110" => BusMuxOut <= BusMuxIn_R14;
38        when "01111" => BusMuxOut <= BusMuxIn_R15;
39        when "10000" => BusMuxOut <= BusMuxIn_HI;
40        when "10001" => BusMuxOut <= BusMuxIn_LO;
41        when "10010" => BusMuxOut <= BusMuxIn_Zhigh;
42        when "10011" => BusMuxOut <= BusMuxIn_Zlow;
43        when "10100" => BusMuxOut <= BusMuxIn_PC;
44        when "10101" => BusMuxOut <= BusMuxIn_MDR;
45        when "10110" => BusMuxOut <= BusMuxIn_InPort;
46        when "10111" => BusMuxOut <= C_sign_extended;
47        when others => BusMuxOut <= (others => '0');
48        end case;
49     end process;
50  end architecture behavioural;
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY multiplexerMDR IS
5      PORT(
6         BusMuxOut, Mdatain: IN STD_LOGIC_VECTOR(31 DOWNTO 0);
7         ReadChannel : IN STD_LOGIC;
8         MDRMuxOut : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
9         );
10  END ENTITY multiplexerMDR;
11
12  ARCHITECTURE behaviour OF multiplexerMDR IS
13  BEGIN
14     MDRMuxOut <= BusMuxOut when (ReadChannel = '0')
15                  else Mdatain when (ReadChannel = '1');
16
17  END ARCHITECTURE behaviour;
```

```
1   library IEEE;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY MUX32toOne IS
5      PORT(
6         R0BusMuxIn,
7         R1BusMuxIn,
8         R2BusMuxIn,
9         R3BusMuxIn,
10        R4BusMuxIn,
11        R5BusMuxIn,
12        R6BusMuxIn,
13        R7BusMuxIn,
14        R8BusMuxIn,
15        R9BusMuxIn,
16        R10BusMuxIn,
17        R11BusMuxIn,
18        R12BusMuxIn,
19        R13BusMuxIn,
20        R14BusMuxIn,
21        R15BusMuxIn,
22        HIBusMuxIn,
23        LOBusMuxIn,
24        ZHIBusMuxIn,
25        ZLOBusMuxIn,
26        PCBusMuxIn,
27        MDRBusMuxIn,
28        INPORTBusMuxIn,
29        C_sign_extended : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
30        s : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
31        BusMuxOut: OUT STD_LOGIC_VECTOR(31 DOWNTO 0));
32  END MUX32toOne;
33
34  ARCHITECTURE behavioral OF MUX32toOne IS
35  BEGIN
36   BigMultiplexer:
37     PROCESS (R0BusMuxIn,
38        R1BusMuxIn,
39        R2BusMuxIn,
40        R3BusMuxIn,
41        R4BusMuxIn,
42        R5BusMuxIn,
43        R6BusMuxIn,
44        R7BusMuxIn,
45        R8BusMuxIn,
46        R9BusMuxIn,
47        R10BusMuxIn,
48        R11BusMuxIn,
49        R12BusMuxIn,
50        R13BusMuxIn,
51        R14BusMuxIn,
52        R15BusMuxIn,
53        HIBusMuxIn,
54        LOBusMuxIn,
55        ZHIBusMuxIn,
56        ZLOBusMuxIn,
57        PCBusMuxIn,
58        MDRBusMuxIn,
59        INPORTBusMuxIn,
60        C_sign_extended,
61        s)
62     BEGIN
```

```
63     CASE s IS
64       when "00000" =>    BusMuxOut <= R0BusMuxIn;
65       when "00001" =>    BusMuxOut <= R1BusMuxIn;
66       when "00010" =>    BusMuxOut <= R2BusMuxIn;
67       when "00011" =>    BusMuxOut <= R3BusMuxIn;
68       when "00100" =>    BusMuxOut <= R4BusMuxIn;
69       when "00101" =>    BusMuxOut <= R5BusMuxIn;
70       when "00110" =>    BusMuxOut <= R6BusMuxIn;
71       when "00111" =>    BusMuxOut <= R7BusMuxIn;
72       when "01000" =>    BusMuxOut <= R8BusMuxIn;
73       when "01001" =>    BusMuxOut <= R9BusMuxIn;
74       when "01010" =>    BusMuxOut <= R10BusMuxIn;
75       when "01011" =>    BusMuxOut <= R11BusMuxIn;
76       when "01100" =>    BusMuxOut <= R12BusMuxIn;
77       when "01101" =>    BusMuxOut <= R13BusMuxIn;
78       when "01110" =>    BusMuxOut <= R14BusMuxIn;
79       when "01111" =>    BusMuxOut <= R15BusMuxIn;
80       when "10000" =>    BusMuxOut <= HIBusMuxIn;
81       when "10001" =>    BusMuxOut <= LOBusMuxIn;
82       when "10010" =>    BusMuxOut <= ZHIBusMuxIn;
83       when "10011" =>    BusMuxOut <= ZLOBusMuxIn;
84       when "10100" =>    BusMuxOut <= PCBusMuxIn;
85       when "10101" =>    BusMuxOut <= MDRBusMuxIn;
86       when "10110" =>    BusMuxOut <= INPORTBusMuxIn;
87       when "10111" =>    BusMuxOut <= C_sign_extended;
88       when others =>     BusMuxOut <= (others => '0');
89
90   END CASE;
91      END PROCESS;
92   END behavioral;
```

```vhdl
library IEEE;
use ieee.std_logic_1164.all;

entity notGate is
   port(
      B        : in std_logic_vector(31 downto 0);
      S        : out std_logic_vector(31 downto 0)
   );
end entity;

architecture behaviour of notGate is

begin
   process(B)
   begin
      for i in 0 to 31 loop
         S(i) <= not B(i);
      end loop;
   end process;
end behaviour;
```

```vhdl
library IEEE;
use ieee.std_logic_1164.all;

entity notGate1bit is
   port(
      B        : in std_logic;
      S        : out std_logic
   );
end entity;

architecture behaviour of notGate1bit is

begin
   S <= not B;
end behaviour;
```

```vhdl
library IEEE;
use ieee.std_logic_1164.all;

entity orGate is
   port(
      A, B     : in std_logic_vector(31 downto 0);
      S        : out std_logic_vector(31 downto 0)
   );
end orGate;

architecture behaviour of orGate is

begin
   process(A, B)
   begin
      for I in 0 to 31 loop
         if(A(I) = '1' or B(I) = '1') then
         S(I) <= '1';
         else
         S(I) <='0';
         end if;
      end loop;
   end process;
end behaviour;
```

```
1   library IEEE;
2   use ieee.std_logic_1164.all;
3
4   entity pcIncrement is
5      port(
6         B     :  in std_logic_vector(31 downto 0);
7         Cin   :  in std_logic;
8         S     :  out std_logic_vector(31 downto 0);
9         Cout  :  out std_logic
10     );
11  end pcIncrement;
12
13  architecture behaviour of pcIncrement is
14     signal increment : std_logic_vector(31 downto 0) := "00000000000000000000000000000001";
15     signal wastedCarry : std_logic;
16
17     component thirtyTwoBitRippleCarryAdder  is
18        port(
19           A     :   in std_logic_vector(31 downto 0);
20           B     :  in std_logic_vector(31 downto 0);
21           Cin   :  in std_logic;
22           S     :  out std_logic_vector(31 downto 0);
23           Cout  :  out std_logic
24        );
25     end component thirtyTwoBitRippleCarryAdder;
26  begin
27
28     U0 : thirtyTwoBitRippleCarryAdder  port map(
29                        A        => increment,
30                        B        => B,
31                        Cin      => Cin,
32                        S        => S,
33                        Cout     => Cout
34                    );
35  end behaviour;
```

```
1   -- megafunction wizard: %RAM: 1-PORT%
2   -- GENERATION: STANDARD
3   -- VERSION: WM1.0
4   -- MODULE: altsyncram
5
6   -- ============================================================
7   -- File Name: ram.vhd
8   -- Megafunction Name(s):
9   --          altsyncram
10  --
11  -- Simulation Library Files(s):
12  --          altera_mf
13  -- ============================================================
14  -- ************************************************************
15  -- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
16  --
17  -- 13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
18  -- ************************************************************
19
20
21  --Copyright (C) 1991-2013 Altera Corporation
22  --Your use of Altera Corporation's design tools, logic functions
23  --and other software and tools, and its AMPP partner logic
24  --functions, and any output files from any of the foregoing
25  --(including device programming or simulation files), and any
26  --associated documentation or information are expressly subject
27  --to the terms and conditions of the Altera Program License
28  --Subscription Agreement, Altera MegaCore Function License
29  --Agreement, or other applicable license agreement, including,
30  --without limitation, that your use is for the sole purpose of
31  --programming logic devices manufactured by Altera and sold by
32  --Altera or its authorized distributors.  Please refer to the
33  --applicable agreement for further details.
34
35
36  LIBRARY ieee;
37  USE ieee.std_logic_1164.all;
38
39  LIBRARY altera_mf;
40  USE altera_mf.all;
41
42  ENTITY ram_n IS
43     PORT
44     (
45        address     : IN STD_LOGIC_VECTOR (8 DOWNTO 0);
46        clock     : IN STD_LOGIC  := '1';
47        data      : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
48        rden      : IN STD_LOGIC  := '1';
49        wren      : IN STD_LOGIC ;
50        q     : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
51     );
52  END ram_n;
53
54
55  ARCHITECTURE SYN OF ram_n IS
56
57     SIGNAL sub_wire0   : STD_LOGIC_VECTOR (31 DOWNTO 0);
58
59
60
61     COMPONENT altsyncram
62     GENERIC (
```

```
63          clock_enable_input_a     : STRING;
64          clock_enable_output_a       : STRING;
65          intended_device_family      : STRING;
66          lpm_hint     : STRING;
67          lpm_type     : STRING;
68          numwords_a      : NATURAL;
69          operation_mode     : STRING;
70          outdata_aclr_a     : STRING;
71          outdata_reg_a      : STRING;
72          power_up_uninitialized      : STRING;
73          read_during_write_mode_port_a     : STRING;
74          widthad_a       : NATURAL;
75          width_a      : NATURAL;
76          width_byteena_a       : NATURAL
77       );
78       PORT (
79             address_a    : IN STD_LOGIC_VECTOR (8 DOWNTO 0);
80             clock0    : IN STD_LOGIC ;
81             data_a    : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
82             wren_a    : IN STD_LOGIC ;
83             q_a    : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
84             rden_a    : IN STD_LOGIC
85       );
86       END COMPONENT;
87
88    BEGIN
89       q    <= sub_wire0(31 DOWNTO 0);
90
91       altsyncram_component : altsyncram
92       GENERIC MAP (
93          clock_enable_input_a => "BYPASS",
94          clock_enable_output_a => "BYPASS",
95          intended_device_family => "Cyclone III",
96          lpm_hint => "ENABLE_RUNTIME_MOD=NO",
97          lpm_type => "altsyncram",
98          numwords_a => 512,
99          operation_mode => "SINGLE_PORT",
100         outdata_aclr_a => "NONE",
101         outdata_reg_a => "CLOCK0",
102         power_up_uninitialized => "FALSE",
103         read_during_write_mode_port_a => "NEW_DATA_NO_NBE_READ",
104         widthad_a => 9,
105         width_a => 32,
106         width_byteena_a => 1
107      )
108      PORT MAP (
109         address_a => address,
110         clock0 => clock,
111         data_a => data,
112         wren_a => wren,
113         rden_a => rden,
114         q_a => sub_wire0
115      );
116
117
118
119   END SYN;
```

```
1   ram_inst : ram PORT MAP (
2         address   => address_sig,
3         clock  => clock_sig,
4         data   => data_sig,
5         rden   => rden_sig,
6         wren   => wren_sig,
7         q   => q_sig
8      );
9
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY reg_32 IS
5      PORT(
6         clk, clr, Rin : IN STD_LOGIC;
7         BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
8         BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
9         );
10  END ENTITY reg_32;
11
12  ARCHITECTURE arc OF reg_32 IS
13  BEGIN
14     SingleRegister:
15     PROCESS (clk, clr, Rin, BusMuxOut)
16     BEGIN
17        IF (rising_edge(clk) AND (Rin = '1')) THEN
18           IF (clr = '0') THEN
19              BusMuxIn <= (others => '0' );
20           ELSE
21              BusMuxIn <= BusMuxOut;
22           END IF;
23        END IF;
24     END PROCESS;
25  END ARCHITECTURE arc;
26
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY reg_IR IS
5      PORT(
6         clk, clr, Rin : IN STD_LOGIC;
7         BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
8         BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
9         );
10  END ENTITY reg_IR;
11
12  ARCHITECTURE arc OF reg_IR IS
13  BEGIN
14     SingleRegister:
15     PROCESS (clk, clr, Rin, BusMuxOut)
16     BEGIN
17        IF (rising_edge(clk) AND (Rin = '1')) THEN
18           IF (clr = '0') THEN
19              BusMuxIn <= (others => '0' );
20           ELSIF(clr = '1' and BusMuxOut(18) = '1') THEN
21              BusMuxIn(17 downto 0) <= BusMuxOut(17 downto 0);
22              BusMuxIn(31 downto 18) <= "11111111111111";
23           ELSE
24              BusMuxIn(17 downto 0) <= BusMuxOut(17 downto 0);
25              BusMuxIn(31 downto 18) <= "00000000000000";
26           END IF;
27        END IF;
28     END PROCESS;
29  END ARCHITECTURE arc;
30
```

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.ALL;
3
4   ENTITY reg_Zero IS
5      PORT(
6         clk, clr, Rin, BAout : IN STD_LOGIC;
7         BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
8         BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
9         );
10  END ENTITY reg_Zero;
11
12  architecture behaviour of reg_Zero is
13
14  signal regToAnd : std_logic_vector(31 downto 0);
15
16  component reg_32 IS
17     PORT(
18        clk, clr, Rin : IN STD_LOGIC;
19        BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
20        BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
21        );
22  END component;
23
24  component andGate_32vs1 is
25     port(
26        A     : in std_logic;
27        B     : in std_logic_vector(31 downto 0);
28        S     : out std_logic_vector(31 downto 0)
29     );
30  end component;
31
32  begin
33  U0: reg_32 port map(
34     clk => clk,
35     clr => clr,
36     Rin => Rin,
37     BusMuxOut => BusMuxOut,
38     BusMuxIn => regToAnd
39  );
40  U1: andGate_32vs1 port map(
41     A => BAout,
42     B => regToAnd,
43     S => BusMuxIn
44  );
45
46  end architecture;
```

```
1   library ieee;
2   USE ieee.std_logic_1164.all;
3
4
5
6   entity registerFile is
7      port
8      (
9         clk, clr,  BAout  : in std_logic;
10        Rin   : in std_logic_vector(15 downto 0);
11        BusMuxOut   : in std_logic_vector(31 downto 0);
12        BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3,
13        BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,
14        BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11,
15        BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15 : out std_logic_vector(31 downto 0)
16     );
17  end registerFile;
18
19  architecture behaviour of registerFile IS
20
21
22  component reg_32
23     port(
24        clk         : in std_logic;
25        clr         : in std_logic;
26        Rin      : in std_logic;
27        BusMuxOut   : in std_logic_vector(31 downto 0);
28        BusMuxIn    : out std_logic_vector(31 downto 0)
29     );
30  end component;
31
32  component reg_Zero IS
33     PORT(
34        clk, clr, Rin, BAout : IN STD_LOGIC;
35        BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
36        BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
37        );
38  END component;
39
40  begin
41  U0: reg_Zero PORT MAP(
42                 clk         => clk,
43                 clr         => clr,
44                 Rin         => Rin(0),
45                 BAout       => BAout,
46                 BusMuxOut   => BusMuxOut,
47                 BusMuxIn    => BusMuxInR0
48              );
49  U1: reg_32 PORT MAP(
50                 clk         => clk,
51                 clr         => clr,
52                 Rin         => Rin(1),
53                 BusMuxOut   => BusMuxOut,
54                 BusMuxIn    => BusMuxInR1
55              );
56  U2: reg_32 PORT MAP(
57                 clk         => clk,
58                 clr         => clr,
59                 Rin         => Rin(2),
60                 BusMuxOut   => BusMuxOut,
61                 BusMuxIn    => BusMuxInR2
62              );
```

```
63   U3: reg_32 PORT MAP(
64                   clk          => clk,
65                   clr          => clr,
66                   Rin          => Rin(3),
67                   BusMuxOut    => BusMuxOut,
68                   BusMuxIn     => BusMuxInR3
69               );
70   U4: reg_32 PORT MAP(
71                   clk          => clk,
72                   clr          => clr,
73                   Rin          => Rin(4),
74                   BusMuxOut    => BusMuxOut,
75                   BusMuxIn     => BusMuxInR4
76               );
77   U5: reg_32 PORT MAP(
78                   clk          => clk,
79                   clr          => clr,
80                   Rin          => Rin(5),
81                   BusMuxOut    => BusMuxOut,
82                   BusMuxIn     => BusMuxInR5
83               );
84   U6: reg_32 PORT MAP(
85                   clk          => clk,
86                   clr          => clr,
87                   Rin          => Rin(6),
88                   BusMuxOut    => BusMuxOut,
89                   BusMuxIn     => BusMuxInR6
90               );
91   U7: reg_32 PORT MAP(
92                   clk          => clk,
93                   clr          => clr,
94                   Rin          => Rin(7),
95                   BusMuxOut    => BusMuxOut,
96                   BusMuxIn     => BusMuxInR7
97               );
98   U8: reg_32 PORT MAP(
99                   clk          => clk,
100                  clr          => clr,
101                  Rin          => Rin(8),
102                  BusMuxOut    => BusMuxOut,
103                  BusMuxIn     => BusMuxInR8
104              );
105  U9: reg_32 PORT MAP(
106                  clk          => clk,
107                  clr          => clr,
108                  Rin          => Rin(9),
109                  BusMuxOut    => BusMuxOut,
110                  BusMuxIn     => BusMuxInR9
111              );
112  U10: reg_32 PORT MAP(
113                  clk          => clk,
114                  clr          => clr,
115                  Rin          => Rin(10),
116                  BusMuxOut    => BusMuxOut,
117                  BusMuxIn     => BusMuxInR10
118              );
119  U11: reg_32 PORT MAP(
120                  clk          => clk,
121                  clr          => clr,
122                  Rin          => Rin(11),
123                  BusMuxOut    => BusMuxOut,
124                  BusMuxIn     => BusMuxInR11
```

```
125                );
126    U12: reg_32 PORT MAP(
127                   clk          => clk,
128                   clr          => clr,
129                   Rin          => Rin(12),
130                   BusMuxOut    => BusMuxOut,
131                   BusMuxIn     => BusMuxInR12
132                );
133    U13: reg_32 PORT MAP(
134                   clk          => clk,
135                   clr          => clr,
136                   Rin          => Rin(13),
137                   BusMuxOut    => BusMuxOut,
138                   BusMuxIn     => BusMuxInR13
139                );
140    U14: reg_32 PORT MAP(
141                   clk          => clk,
142                   clr          => clr,
143                   Rin          => Rin(14),
144                   BusMuxOut    => BusMuxOut,
145                   BusMuxIn     => BusMuxInR14
146                );
147    U15: reg_32 PORT MAP(
148                   clk          => clk,
149                   clr          => clr,
150                   Rin          => Rin(15),
151                   BusMuxOut    => BusMuxOut,
152                   BusMuxIn     => BusMuxInR15
153                );
154    end behaviour;
155
```

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY regMAR IS
   PORT(
      clk, clr, Rin : IN STD_LOGIC;
      BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
      BusMuxIn : OUT STD_LOGIC_VECTOR(8 DOWNTO 0)
      );
END ENTITY regMAR;

ARCHITECTURE arc OF regMAR IS
BEGIN
   SingleRegister:
   PROCESS (clk, clr, Rin, BusMuxOut)
   BEGIN
      IF (rising_edge(clk) AND (Rin = '1')) THEN
         IF (clr = '0') THEN
            BusMuxIn <= (others => '0' );
         ELSE
            BusMuxIn <= BusMuxOut(8 downto 0);
         END IF;
      END IF;
   END PROCESS;
END ARCHITECTURE arc;
```

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY regPC IS
   PORT(
      clk, clr, Rin : IN STD_LOGIC;
      BusMuxOut : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
      BusMuxIn : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
      );
END ENTITY regPC;

ARCHITECTURE arc OF regPC IS
BEGIN
   SingleRegister:
   PROCESS (clk, clr, Rin, BusMuxOut)
   BEGIN
      IF (clr = '0') THEN
         BusMuxIn <= (others => '0' );
      Elsif ((clr = '0') and rising_edge(clk) AND (Rin = '1')) THEN
         BusMuxIn <= BusMuxOut;
      END IF;
   END PROCESS;
END ARCHITECTURE arc;

```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity selectAndEncodeLogic is
5     port(
6        IRin                              : in std_logic_vector(31 downto 0);
7        Gra, Grb, Grc, Rin, Rout, BAout   : in std_logic;
8        BusMuxOut                         : in std_logic_vector(31 downto 0);
9        C_sign_extended                   : out std_logic_vector(31 downto 0);
10       r0in_r15in_Decoded                : out std_logic_vector(15 downto 0);
11       r0out_r15out_Decoded              : out std_logic_vector(15 downto 0)
12    );
13 end entity;
14
15 architecture behaviour of selectAndEncodeLogic is
16
17 component selectAndEncodeSubComponent1 is
18    port(
19       IRin           : in std_logic_vector(31 downto 0);
20       Gra, Grb, Grc  : in std_logic;
21       output         : out std_logic_vector(3 downto 0)
22    );
23 end component;
24 component selectAndEncodeSubComponent2 is
25    port(
26       input                   : in std_logic_vector(15 downto 0);
27       Rin, Rout, BAout        : in std_logic;
28       r0in_r15in_Decoded      : out std_logic_vector(15 downto 0);
29       r0out_r15out_Decoded    : out std_logic_vector(15 downto 0)
30    );
31 end component;
32 component selectAndEncodeSubComponent3 is
33    port(
34       input                : in std_logic_vector(31 downto 0);
35       output               : out std_logic_vector(31 downto 0)
36    );
37 end component;
38 component decoder16bits is
39    port (
40       input  : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
41       output : OUT STD_LOGIC_VECTOR(15 downto 0)
42    );
43 end component;
44
45
46 signal decoderInput  : std_logic_vector(3 downto 0);
47 signal decoderOutput : std_logic_vector(15 downto 0);
48
49 begin
50
51 U0: selectAndEncodeSubComponent1 port map(
52       IRin  => IRin,
53       Gra   => Gra,
54       Grb   => Grb,
55       Grc   => Grc,
56       output=> decoderInput
57
58 );
59 U1: decoder16bits port map(
60       input  => decoderInput,
61       output => decoderOutput
62 );
```

```
63   U2: selectAndEncodeSubComponent2  port map(
64         input                => decoderOutput,
65         Rin                  => Rin,
66         Rout                 => Rout,
67         BAout                => BAout,
68         r0in_r15in_Decoded   => r0in_r15in_Decoded,
69         r0out_r15out_Decoded => r0out_r15out_Decoded
70   );
71   U3: selectAndEncodeSubComponent3  port map(
72         input => IRin,
73         output => C_sign_extended
74   );
75   end architecture;
76
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity selectAndEncodeSubComponent1  is
   port(
      IRin                        : in std_logic_vector(31 downto 0);
      Gra, Grb, Grc               : in std_logic;
      output                      : out std_logic_vector(3 downto 0)
   );
end entity;

architecture behaviour of selectAndEncodeSubComponent1  is
begin
--The IR register is split into the following fields
--5 bits for opcode
--4 bits for Ra
--4 bits for Rb
--4 bits for Rc
output <= IRin(26 downto 23) when (Gra = '1' and Grb = '0' and Grc = '0')
          else IRin(22 downto 19) when (Gra = '0' and Grb = '1' and Grc = '0')
          else IRin(18 downto 15) when (Gra = '0' and Grb = '0' and Grc = '1');
end architecture;

--CONFIRMED TO WORK
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity selectAndEncodeSubComponent2  is
5      port(
6         input                    : in std_logic_vector(15 downto 0);
7         Rin, Rout, BAout         : in std_logic;
8         r0in_r15in_Decoded       : out std_logic_vector(15 downto 0);
9         r0out_r15out_Decoded     : out std_logic_vector(15 downto 0)
10     );
11  end entity;
12
13  architecture behaviour of selectAndEncodeSubComponent2  is
14  begin
15     process(Rin, input, BAout, Rout)
16     begin
17        for I in 0 to 15 loop
18           r0in_r15in_Decoded(I)  <= input(I) and Rin;
19           r0out_r15out_Decoded(I)<=input(I) and (BAout or Rout);
20        end loop;
21     end process;
22  end architecture;
23
24  --CONFIRMED TO WORK
```

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   library work;
5   entity selectAndEncodeSubComponent3  is
6   port(
7      input                : in std_logic_vector(31 downto 0);
8      output               : out std_logic_vector(31 downto 0)
9   );
10  end entity;
11
12  architecture behaviour of selectAndEncodeSubComponent3  is
13
14  begin
15
16  --control of extention is on 0x00040000 (3rd bit of 4th "group")
17  output(31 downto 18) <= (others => input(18));
18  output(17 downto 0) <= input(17 downto 0);
19  end behaviour;
20
21  --CONFIRMED TO WORK
```