

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 entity top_level_tb is
6 end entity;
7
8 architecture behavior of top_level_tb is
9     constant TIME_DELAY : time := 10 ns;
10    constant NUM_VALS : integer := 26;
11
12    type RegWr_array is array (0 to (NUM_VALS - 1)) of std_logic;
13    type Rd_array is array (0 to (NUM_VALS - 1)) of std_logic_vector(4 downto 0);
14    type Rs_array is array (0 to (NUM_VALS - 1)) of std_logic_vector(4 downto 0);
15    type Rt_array is array (0 to (NUM_VALS - 1)) of std_logic_vector(4 downto 0);
16    type ALUctr_array is array (0 to (NUM_VALS - 1)) of std_logic_vector(2 downto 0);
17    type Zero_array is array (0 to (NUM_VALS - 1)) of std_logic;
18    type Overflow_array is array(0 to (NUM_VALS - 1)) of std_logic;
19    type Carryout_array is array(0 to (NUM_VALS - 1)) of std_logic;
20    type Result_array is array (0 to (NUM_VALS - 1)) of std_logic_vector(31 downto 0);
21
22    -- Expected input and output data.
23    constant RegWr_vals : RegWr_array :=
24        ('0','1','0','1','0','1','0','1','0','1','0','1','0','1','0','1','0','1','0','1','0','1','0','1','0','1');
25    constant Rd_vals : Rd_array :=
26        ("00011","00011","00011","00011","00011","00011","00111","00111","00100","00100","00101",
27         "00101","00110","00110","00111","00111","01000","01000","01001","01001","01010","01010",
28         "01011","01011","01100","01100");
29    constant Rs_vals : Rs_array :=
30        ("00001","00001","00001","00001","00001","00001","00111","00111","00010","00010","00011",
31         "00011","00101","00101","00110","00110","00011","00011","01000","01000","00110","00110",
32         "01000","01000","01000","01000");
33    constant Rt_vals : Rt_array :=
34        ("00010","00010","00010","00010","00010","00010","00111","00111","00011","00011","00100",
35         "00100","00101","00101","00101","00101","00101","00101","00101","00101","00101","00111","00111",
36         "01000","01000","01000","01000");
37    constant ALUctr_vals : ALUctr_array := ("011","011","000","000","001",
38        "001","000","000","010","010","000","000","100","100","110","110","111","111","101","101",
39        "000","000","000","000","001","001");
40    constant Zero_vals : Zero_array :=
41        ('0','0','0','0','0','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1',
42         '1','1','1','1');
43    constant Overflow_vals : Overflow_array :=
44        ('0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0',
45         '0','0','0','0');
46    constant Carryout_vals : Carryout_array :=
47        ('0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','1','1','1','1','1','1','1','0','0',
48         '1','1','1','1');
49    constant Result_vals : Result_array := ("00000000000000000000000000000011",
50        "00000000000000000000000000000011",
51        "00000000000000000000000000000011",
52        "00000000000000000000000000000011",
53        "11111111111111111111111111111111",
54        "11111111111111111111111111111111",
55        "00000000000000000000000000000000",
56        "00000000000000000000000000000000",
57        "00000000000000000000000000000010",
58        "00000000000000000000000000000010",
59        "00000000000000000000000000000001",
60        "00000000000000000000000000000001",
61        "00000000000000000000000000000010",
62        "00000000000000000000000000000010",
63        "000000000000000000000000000000100",
64        "000000000000000000000000000000100",
65        "11111111111111111111111111111111",
66        "11111111111111111111111111111111",
67        "01111111111111111111111111111111",
68        "01111111111111111111111111111111")

```

```

51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
--signal clk_sig : std_logic := '0';
signal RegWr_sig : std_logic;
signal Rd_sig : std_logic_vector(4 downto 0);
signal Rs_sig : std_logic_vector(4 downto 0);
signal Rt_sig : std_logic_vector(4 downto 0);
signal ALUctr_sig : std_logic_vector(2 downto 0);
signal Zero_sig : std_logic;
signal Overflow_sig : std_logic;
signal Carryout_sig : std_logic;
signal Result_sig : std_logic_vector(31 downto 0) := "00000000000000000000000000000000";

begin

DUT : entity work.top_level(simple)
  port map(--clk => clk_sig,
    RegWr => RegWr_sig,
    Rd => Rd_sig,
    Rs => Rs_sig,
    Rt => Rt_sig,
    ALUctr => ALUctr_sig,
    Zero => Zero_sig,
    Overflow => Overflow_sig,
    Carryout => Carryout_sig,
    Result => Result_sig);

stimulus : process
begin
  for i in 0 to (NUM_VALS - 1) loop
    RegWr_sig <= RegWr_vals(i);
    Rd_sig <= Rd_vals(i);
    Rs_sig <= Rs_vals(i);
    Rt_sig <= Rt_vals(i);
    ALUctr_sig <= ALUctr_vals(i);
    wait for TIME_DELAY;
  end loop;
  wait;
end process stimulus;

monitor : process
  variable i : integer := 0;
begin
  wait for TIME_DELAY/4;
  while (i < NUM_VALS) loop
    assert RegWr_sig = RegWr_vals(i)
      report "RegWr value is incorrect."
      severity note;

    assert Rd_sig = Rd_vals(i)
      report "Rd value is incorrect."
      severity note;

    assert Rs_sig = Rs_vals(i)
      report "Rs value is incorrect."
      severity note;

    assert Rt_sig = Rt_vals(i)
      report "Rt value is incorrect."
      severity note;

    assert ALUctr_sig = ALUctr_vals(i)
      report "ALUctr value is incorrect."

```

```
120         severity note;
121
122     wait for TIME_DELAY/2;
123
124     assert Zero_sig = Zero_vals(i)
125         report "Zero value is incorrect."
126         severity note;
127
128     assert Overflow_sig = Overflow_vals(i)
129         report "Overflow value is incorrect."
130         severity note;
131
132     assert Carryout_sig = Carryout_vals(i)
133         report "Carryout value is incorrect."
134         severity note;
135
136     assert Result_sig = Result_vals(i)
137         report "Results value is incorrect."
138         severity note;
139
140     i := i + 1;
141     wait for TIME_DELAY/2;
142 end loop;
143 wait;
144 end process monitor;
145
146 end behavior;
147
```