```
1   `timescale 1ps/1ps
2   module tb_RegFile_wrapper();
3
4       reg [9:0] data_input;
5       reg ld_Reg, ld_Setup, ld_Imm, clk, en;
6
7       wire [4:0] Flags;
8       wire [6:0] out1, out2, out3, out4;
9       wire[15:0] RdestOut;
10
11
12      RegFile_wrapper uut (
13          .data_input(data_input),
14          .ld_Reg(ld_Reg),
15          .ld_Setup(ld_Setup),
16          .ld_Imm(ld_Imm),
17          .clk(clk),
18          .en(en),
19          .Flags(Flags),
20          .out1(out1),
21          .out2(out2),
22          .out3(out3),
23          .out4(out4),
24          .RdestOut(RdestOut)
25      );
26
27      initial begin
28          //initialize the clock
29          clk = 0;
30          en = 0;
31
32
33          //resets all registers to 0
34          data_input = 8'b00000000;
35          ld_Setup = 1;
36          #5; //clk = 1
37          ld_Setup = 0;
38          #5;//clk = 0
39          data_input = 8'b10000000;
40          ld_Setup = 1;
41          #5; //clk = 0
42          ld_Setup = 0;
43          #10; //clk = 0
44
45
46          //choose a register for rdest and rsrc
47          data_input = 8'b00000001; //Rdest = reg0; Rsrc = reg1
48          ld_Reg = 1;
49          #5; //clk = 1
50          ld_Reg = 0;
51          #5; //clk = 0
52
53
54          //set immediate value
55          data_input = 1;
56          ld_Imm = 1;
57          #5; //clk = 1
58          ld_Imm = 0;
59          #5 //clk = 0
60
61
62          //prepare to load the immediate
63          data_input = 8'b10010000;
64          ld_Setup = 1;
65          #5; //clk = 1
66          ld_Setup = 0;
67          #5; //clk = 0
68
69
70          en = 1;
71          #10; //clk = 0
72          en = 0;
73
```

```
74
75
76
77
78
79
80
81
82    //    clk = 1;
83    //    $display("Starting Regfile + ALU Wrapper Testbench");
84    //
85    //    data_input = 10'b0001000000;
86    //    ld_Reg = 0;
87    //    #10;
88    //
89    //    ld_Reg = 1;
90    //    data_input = 10'b0000000000;
91    //    ld_Setup = 0;
92    //    #10;
93    //
94    //    ld_Setup = 1;
95    //    data_input = 10'b1111000000;
96    //    ld_Imm = 0;
97    //    #10;
98    //
99    //    ld_Imm = 1;
100   //    ld_Inst = 0;
101   //    #10;
102
103
104         //rst all registers
105         /*
106         data_input = 10'b1000000000;
107         ld_Setup = 1;
108         ld_clk = 1;
109         ld_Imm = 1;
110         ld_Reg = 1;
111
112         #5;
113         ld_Setup = 0;
114
115         #5;
116         ld_Setup = 1;
117
118
119
120          #10;
121         //simulate actual user input ADD 8 to the first register
122         data_input = 10'b0001000000;
123         ld_Reg = 0;
124
125         #10;
126
127         ld_Reg = 1;
128         data_input = 10'b0110000000;
129         ld_Setup = 0;
130
131         #10;
132
133         ld_Setup = 1;
134         data_input = 10'b0100000000;
135         ld_Imm = 0;
136
137         #10;
138         ld_Imm = 0;
139         ld_clk = 0;
140
141          #10;
142         ld_clk = 1;
143
144         //step through again so its on the pos edge
145         ld_clk = 0;
146
```

```
147          #10;
148         ld_clk = 1;
149         #10;
150         ld_clk = 0;
151         #10;
152         ld_clk = 1;
153         #10;
154         ld_clk = 0;
155         #10;
156         */
157
158      end
159      always #5 clk = ~clk;
160
161
162   endmodule
163
```