```
1
2   /*
3      R0_R15_GenPurposeRegs is an abstraction layer for reg_file.v, given the control signals
    required in the project specification.
4      It encodes the 16 one-hot-encoded enable and 'read' signals as 4-bit write and read
    addresses to the register file.
5   */
6
7   /*
8      DESIGN DECISIONS:
9      We chose to *vectorize* the enable and read signals to provide more clarity to our design.
10     Therefore, in all instantiations of this module, signals like 'rXin' are grouped as a
    single vector GRin.
11     Similarly, signals like 'rXout' are grouped as a single vector GRout.
12
13     PHASE 2 EDIT:
14     When BAout is high and the address presented to the module refers to R0,
15     zero is placed on the output data lines rather than the contents of the register.
16  */
17
18  module R0_R15_GenPurposeRegs (
19     input clk, reg_clear, BAout,
20     input [31:0] BusMuxOut,
21     input [15:0] GRin, // enable vector (One-Hot). IN refers to the perspective of the
    registers, not the Bus.
22     input [15:0] GRout, // read vector (One-Hot). OUT refers to the perspective of the
    registers, not the Bus.
23
24     output [31:0] BusMuxIn
25     );
26
27     wire [3:0] w_addr; // Encoded write address
28     wire [3:0] r_addr; // Encoded read addresses
29     wire [31:0] w_data; // Write data from Bus.
30     wire enable;
31     wire [31:0] data_out;
32
33
34     //Encode 16 r..in signals to w_addr
35
36     assign w_addr[0] = GRin[1] | GRin[3] | GRin[5] | GRin[7] | GRin[9] | GRin[11] | GRin[13]
    | GRin[15];
37     assign w_addr[1] = GRin[2] | GRin[3] | GRin[6] | GRin[7] | GRin[10] | GRin[11] | GRin[14]
     | GRin[15];
38     assign w_addr[2] = GRin[4] | GRin[5] | GRin[6] | GRin[7] | GRin[12] | GRin[13] | GRin[14]
     | GRin[15];
39     assign w_addr[3] = GRin[8] | GRin[9] | GRin[10] | GRin[11] | GRin[12] | GRin[13] | GRin[
    14] |GRin[15];
40
41     //Encode 16 r..out signals to r_addr
42
43     assign r_addr[0] = GRout[1] | GRout[3] | GRout[5] | GRout[7] | GRout[9] | GRout[11] |
    GRout[13] | GRout[15];
44     assign r_addr[1] = GRout[2] | GRout[3] | GRout[6] | GRout[7] | GRout[10] | GRout[11] |
    GRout[14] | GRout[15];
45     assign r_addr[2] = GRout[4] | GRout[5] | GRout[6] | GRout[7] | GRout[12] | GRout[13] |
    GRout[14] | GRout[15];
46     assign r_addr[3] = GRout[8] | GRout[9] | GRout[10] | GRout[11] | GRout[12] | GRout[13] |
    GRout[14] |GRout[15];
47
48     //BusMuxOut is w_data
49
50     assign w_data = BusMuxOut;
51
52     // Enable logic: clear or any r..in signal
53     // Using Reduction or to check if any value in encoded signal w_addr is 1
54
55     assign enable = GRin[0] | (|w_addr);
56
57     // 16x32reg_file Module
58     reg_file RF(clk, reg_clear, enable, r_addr, w_addr, w_data, data_out);
59
60     assign BusMuxIn = (GRout[0] && BAout) ? 32'b0 : data_out;
61
62  endmodule
63
```

64
65