```
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     Furthermore, this module incorporates two read ports in case we opt for a 3-bus design
    at a later phase.
14  */
15
16  module R0_R15_GenPurposeRegs #(
17     parameter ClrVal = 32'b0
18     )(
19     input clk, reg_clear,
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] GRoutA, // read vector (One-Hot). OUT refers to the perspective of the
    registers, not the Bus.
23
24     output [31:0] BusMuxIn,
25     output [31:0] BusMuxIn2
26     );
27
28     wire [3:0] w_addr; // Encoded write address
29     wire [3:0] r_addrA; // Encoded read addresses
30     wire [3:0] r_addrB;
31     wire [31:0] w_data; // Write data from Bus.
32     wire enable;
33
34
35     //Encode 16 r..in signals to w_addr
36
37     assign w_addr[0] = GRin[1] | GRin[3] | GRin[5] | GRin[7] | GRin[9] | GRin[11] | GRin[13]
    | GRin[15];
38     assign w_addr[1] = GRin[2] | GRin[3] | GRin[6] | GRin[7] | GRin[10] | GRin[11] | GRin[14]
     | GRin[15];
39     assign w_addr[2] = GRin[4] | GRin[5] | GRin[6] | GRin[7] | GRin[12] | GRin[13] | GRin[14]
     | GRin[15];
40     assign w_addr[3] = GRin[8] | GRin[9] | GRin[10] | GRin[11] | GRin[12] | GRin[13] | GRin[
    14] |GRin[15];
41
42     //Encode 16 r..out signals to r_addr
43
44     assign r_addrA[0] = GRoutA[1] | GRoutA[3] | GRoutA[5] | GRoutA[7] | GRoutA[9] | GRoutA[11
    ] | GRoutA[13] | GRoutA[15];
45     assign r_addrA[1] = GRoutA[2] | GRoutA[3] | GRoutA[6] | GRoutA[7] | GRoutA[10] | GRoutA[
    11] | GRoutA[14] | GRoutA[15];
46     assign r_addrA[2] = GRoutA[4] | GRoutA[5] | GRoutA[6] | GRoutA[7] | GRoutA[12] | GRoutA[
    13] | GRoutA[14] | GRoutA[15];
47     assign r_addrA[3] = GRoutA[8] | GRoutA[9] | GRoutA[10] | GRoutA[11] | GRoutA[12] | GRoutA
    [13] | GRoutA[14] |GRoutA[15];
48
49     // (To be edited if 3-bus design)
50     assign r_addrB = r_addrA;
51
52     // Mux BusMuxOut with default value for clear
53
54     assign w_data = reg_clear ? ClrVal : BusMuxOut;
55
56     // Enable logic: clear or any r..in signal
57     // Using Reduction or to check if any value in encoded signal w_addr is 1
58
59     assign enable = reg_clear | GRin[0] | (|w_addr);
60
```

```
61      // 16x32reg_file Module
62      reg_file RF(clk, enable, r_addrA, r_addrB, w_addr, w_data, BusMuxIn, BusMuxIn2);
63
64  endmodule
65
66
67
```