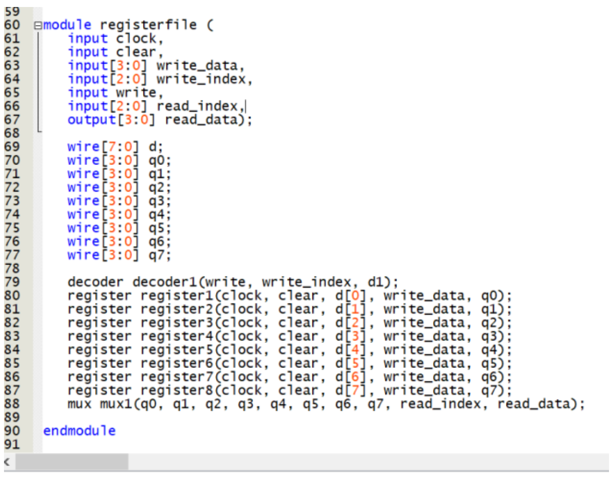
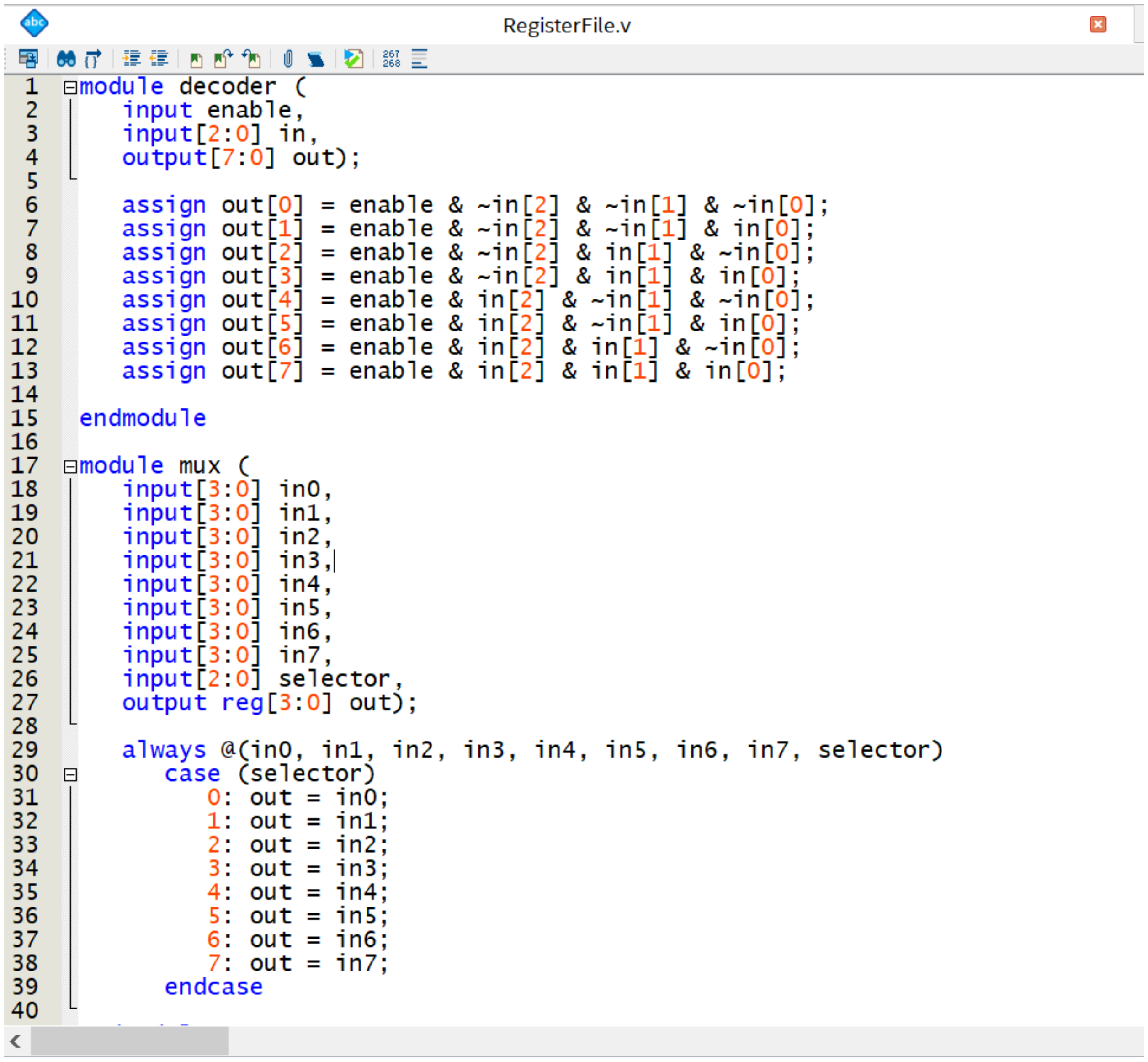
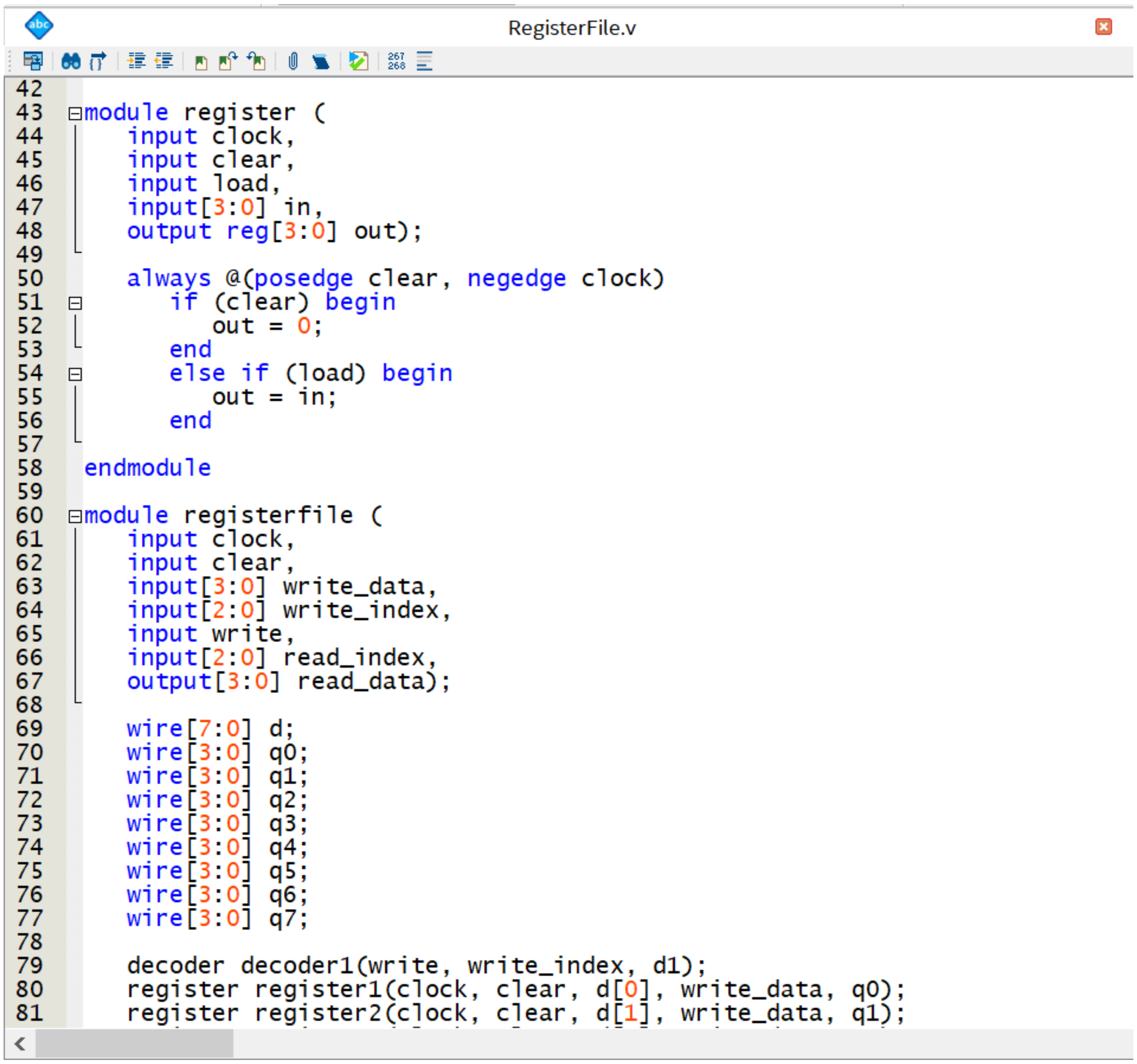


```verilog

module registerfile (
    input clock,
    input clear,
    input[3:0] write_data,
    input[2:0] write_index,
    input write,
    input[2:0] read_index,
    output[3:0] read_data);

    wire[7:0] d;
    wire[3:0] q0;
    wire[3:0] q1;
    wire[3:0] q2;
    wire[3:0] q3;
    wire[3:0] q4;
    wire[3:0] q5;
    wire[3:0] q6;
    wire[3:0] q7;

    decoder decoder1(write, write_index, d1);
    register register1(clock, clear, d[0], write_data, q0);
    register register2(clock, clear, d[1], write_data, q1);
    register register3(clock, clear, d[2], write_data, q2);
    register register4(clock, clear, d[3], write_data, q3);
    register register5(clock, clear, d[4], write_data, q4);
    register register6(clock, clear, d[5], write_data, q5);
    register register7(clock, clear, d[6], write_data, q6);
    register register8(clock, clear, d[7], write_data, q7);
    mux mux1(q0, q1, q2, q3, q4, q5, q6, q7, read_index, read_data);

endmodule

```



RegisterFile.v

```verilog
module decoder (
    input enable,
    input[2:0] in,
    output[7:0] out);

    assign out[0] = enable & ~in[2] & ~in[1] & ~in[0];
    assign out[1] = enable & ~in[2] & ~in[1] & in[0];
    assign out[2] = enable & ~in[2] & in[1] & ~in[0];
    assign out[3] = enable & ~in[2] & in[1] & in[0];
    assign out[4] = enable & in[2] & ~in[1] & ~in[0];
    assign out[5] = enable & in[2] & ~in[1] & in[0];
    assign out[6] = enable & in[2] & in[1] & ~in[0];
    assign out[7] = enable & in[2] & in[1] & in[0];

endmodule

module mux (
    input[3:0] in0,
    input[3:0] in1,
    input[3:0] in2,
    input[3:0] in3,
    input[3:0] in4,
    input[3:0] in5,
    input[3:0] in6,
    input[3:0] in7,
    input[2:0] selector,
    output reg[3:0] out);

    always @(in0, in1, in2, in3, in4, in5, in6, in7, selector)
       case (selector)
          0: out = in0;
          1: out = in1;
          2: out = in2;
          3: out = in3;
          4: out = in4;
          5: out = in5;
          6: out = in6;
          7: out = in7;
       endcase

```



```

module register (
    input clock,
    input clear,
    input load,
    input[3:0] in,
    output reg[3:0] out);

    always @(posedge clear, negedge clock)
       if (clear) begin
          out = 0;
       end
       else if (load) begin
          out = in;
       end

endmodule

module registerfile (
    input clock,
    input clear,
    input[3:0] write_data,
    input[2:0] write_index,
    input write,
    input[2:0] read_index,
    output[3:0] read_data);

    wire[7:0] d;
    wire[3:0] q0;
    wire[3:0] q1;
    wire[3:0] q2;
    wire[3:0] q3;
    wire[3:0] q4;
    wire[3:0] q5;
    wire[3:0] q6;
    wire[3:0] q7;

    decoder decoder1(write, write_index, d1);
    register register1(clock, clear, d[0], write_data, q0);
    register register2(clock, clear, d[1], write_data, q1);
```