```
module logic(
    input [4:0] inputs, // 5 input signals
    output Y
);

wire [3:0] decoder_output;
wire [7:0] minterm_outputs;

// Instantiate the decoder module
decoder4_2 dec(
    .in(inputs[1:0]),
    .out(decoder_output)
);

// Instantiate the 8-to-1 multiplexer module
mux8_1 mux(
    .select(decoder_output),
    .in({
        // Define the minterm outputs here
        // Connect each minterm to the corresponding input combination
        minterm_outputs[0],  // minterm 0
        minterm_outputs[1],  // minterm 1
        minterm_outputs[6],  // minterm 2
        minterm_outputs[9],  // minterm 3
        minterm_outputs[15], // minterm 4
        minterm_outputs[16], // minterm 5
        minterm_outputs[17], // minterm 6
        minterm_outputs[18], // minterm 7
        minterm_outputs[20], // minterm 8
        minterm_outputs[25], // minterm 9
        minterm_outputs[28], // minterm 10
        minterm_outputs[29], // minterm 11
        minterm_outputs[30], // minterm 12
        minterm_outputs[31], // minterm 13
        minterm_outputs[22], // minterm 14
        minterm_outputs[23]  // minterm 15
    }),
    .out(Y)
);

// Define minterm outputs using appropriate logic gates
assign minterm_outputs[0] = inputs[0] & inputs[1] & ~inputs[2] & inputs[3] & inputs[4];
// Define other minterm outputs similarly

endmodule



module decoder4_2(a,b,en,y0,y1,y2,y3);
output y0,y1,y2,y3;
input a, b, en;
```

```
assign y0 = (en&(~a&~b));

assign y1 = (en&(a&~b));

assign y2 = (en&(~a&b));

assign y3 = (en&(a&b));


endmodule


module oneB(

output y,

input w1, w2, w3


);



assign y = (~w1 & (w2&w3))|(w1 & (w2|w3));


endmodule


module bit_shifter(

   input w3, w2, w1, w0, Shift,

   output reg y3, y2, y1, y0, k

);
```

```
always @* begin
    if (Shift) begin
        y3 = 0;
        y2 = w3;
        y1 = w2;
        y0 = w1;
        k = w0;
    end
    else if (!Shift) begin
        y3 = w3;
        y2 = w2;
        y1 = w1;
        y0 = w0;
        k = 0;
    end
    else begin
        y3 = w3;
        y2 = w2;
        y1 = w1;
        y0 = w0;
        k = w0;
    end
end

endmodule
```

```
module mux8_1(

   input a, b, c, d,e,f,g,h ,S0, S1,S2,

   output  y6

);




//mux4_1
assign y0 = (~S0 & a) | (S0 & b);


assign y1 = (~S1 & c) | (S1 & d);


assign y2 = ((S0 ^ S1) & y0) | ((~(S0 ^ S1)) & y1);
//mux4_1_(2)
assign y3 = (~S0 & e) | (S0 & f);


assign y4 = (~S1 & g) | (S1 & h);


assign y5 = ((S0 ^ S1) & y3) | ((~(S0 ^ S1)) & y4);
assign y6 = (y2&S2)^(y5&~S2);


endmodule
```