```verilog
module interleaver(
    input  [7:0] byte0,
    input  [7:0] byte1,
    input  [7:0] byte2,
    input  [7:0] byte3,
    output [7:0] out0,
    output [7:0] out1,
    output [7:0] out2,
    output [7:0] out3
    );

    assign out0 = { byte3[1], byte3[0], byte2[1], byte2[0], byte1[1], byte1[0], byte0[1], byte0[0] };
    assign out1 = { byte3[3], byte3[2], byte2[3], byte2[2], byte1[3], byte1[2], byte0[3], byte0[2] };
    assign out2 = { byte3[5], byte3[4], byte2[5], byte2[4], byte1[5], byte1[4], byte0[5], byte0[4] };
    assign out3 = { byte3[7], byte3[6], byte2[7], byte2[6], byte1[7], byte1[6], byte0[7], byte0[6] };

endmodule

// Note they are the same!
module deinterleaver(
    input  [7:0] byte0,
    input  [7:0] byte1,
    input  [7:0] byte2,
    input  [7:0] byte3,
    output [7:0] out0,
    output [7:0] out1,
    output [7:0] out2,
    output [7:0] out3
    );

    assign out0 = { byte3[1], byte3[0], byte2[1], byte2[0], byte1[1], byte1[0], byte0[1], byte0[0] };
    assign out1 = { byte3[3], byte3[2], byte2[3], byte2[2], byte1[3], byte1[2], byte0[3], byte0[2] };
    assign out2 = { byte3[5], byte3[4], byte2[5], byte2[4], byte1[5], byte1[4], byte0[5], byte0[4] };
    assign out3 = { byte3[7], byte3[6], byte2[7], byte2[6], byte1[7], byte1[6], byte0[7], byte0[6] };

endmodule
```

```verilog
module problem2 #(parameter SIGNED_INPUT = 0, parameter NONE_OF_THEM_ARE_ONE = 0) (
    input     [7:0] problem_a,
    output    [3:0] solution_a,

    input    [15:0] problem_b_1,
    input    [15:0] problem_b_2,
    output   [16:0] solution_b,

    input     [1:0] problem_c_0,
    input     [1:0] problem_c_1,
    input     [1:0] problem_c_2,
    input     [1:0] problem_c_3,
    output    [1:0] solution_c,

    input     [1:0] problem_d_0,
    input     [1:0] problem_d_1,
    input     [1:0] problem_d_2,
    input     [1:0] problem_d_3,
    output reg [1:0] solution_d
    );

    generate
        if (SIGNED_INPUT == 0)
        begin
            assign  solution_a = (problem_a >> 4);
        end
        else
        begin
            assign  solution_a = (problem_a >>> 4);
        end
    endgenerate

    assign solution_b = problem_b_1 + problem_b_2;

    assign solution_c = (problem_c_3 == 1) ? (2'd3) : ( (problem_c_2 == 1) ? (2'd2) : ( (problem_c_1 == 1) ? (2'd1) : ( (problem_c_0 == 1) ? (2'd0) : (NONE_OF_THEM_ARE_ONE) ) ) );

    wire [3:0] is_one;
    assign is_one = {(problem_d_3 == 1), ((problem_d_2 == 1) & ~(problem_d_3 == 1)), ((problem_d_1 == 1) & ~(problem_d_3 == 1) & ~(problem_d_2 == 1)), ((problem_d_0 == 1) & ~(problem_d_3 == 1) & ~(problem_d_2 == 1) & ~(problem_d_1 == 1))};


    localparam I3_IS_ONE = 4'b1000;
    localparam I2_IS_ONE = 4'b0100;
    localparam I1_IS_ONE = 4'b0010;
    localparam I0_IS_ONE = 4'b0001;

    always @(*)
    begin
        case (is_one)
            I3_IS_ONE: solution_d = 2'd3;
            I2_IS_ONE: solution_d = 2'd2;
            I1_IS_ONE: solution_d = 2'd1;
            I0_IS_ONE: solution_d = 2'd0;
            default: solution_d = NONE_OF_THEM_ARE_ONE;
        endcase
    end

endmodule
```

```verilog
module problem3 #(parameter WIDTH = 16) (
    input [WIDTH-1:0] data_in,
    output even_parity_out
    );

    assign even_parity_out = ~^(data_in);
endmodule
```

```
module problem1_tb;
	wire  [7:0] byte0;
	wire  [7:0] byte1;
	wire  [7:0] byte2;
	wire  [7:0] byte3;
	wire  [7:0] out0;
	wire  [7:0] out1;
	wire  [7:0] out2;
	wire  [7:0] out3;

	assign byte0 = 8'h00;
	assign byte1 = 8'h0E;
	assign byte2 = 8'h8C;
	assign byte3 = 8'h03;

	interleaver i1 (
						.byte0(byte0),
						.byte1(byte1),
						.byte2(byte2),
						.byte3(byte3),
						.out0(out0),
						.out1(out1),
						.out2(out2),
						.out3(out3)
				);

	initial
	begin
	   $dumpfile("test.vcd");
		$dumpvars(0,problem1_tb);
		#10;
	end

endmodule
```

```
module problem2_tb;

	wire [7:0] problem_a;
	wire [3:0] solution_a;
	wire [3:0] solution_a2;
	wire [15:0] problem_b_1;
	wire [15:0] problem_b_2;
	wire [16:0] solution_b;
	wire [16:0] solution_b2;
	wire [1:0] problem_c_0;
	wire [1:0] problem_c_1;
	wire [1:0] problem_c_2;
	wire [1:0] problem_c_3;
	wire [1:0] solution_c;
	wire [1:0] solution_c2;
	wire [1:0] problem_d_0;
	wire [1:0] problem_d_1;
	wire [1:0] problem_d_2;
	wire [1:0] problem_d_3;
	wire [1:0] solution_d;
	wire [1:0] solution_d2;

	reg [7:0]  op1;
	reg [15:0] op2a;
	reg [15:0] op2b;

	reg [1:0] i0;
	reg [1:0] i1;
	reg [1:0] i2;
	reg [1:0] i3;

	assign problem_a = op1;
	assign problem_b_1 = op2a;
	assign problem_b_2 = op2b;
	assign problem_c_0 = i0;
	assign problem_c_1 = i1;
	assign problem_c_2 = i2;
	assign problem_c_3 = i3;
	assign problem_d_0 = i0;
	assign problem_d_1 = i1;
	assign problem_d_2 = i2;
	assign problem_d_3 = i3;

	problem2 #(.SIGNED_INPUT(0)) p2a(
                                                                .problem_a(problem_a),
                                                                .solution_a(solution_a),
                                                                .problem_b_1(problem_b_1),
                                                                .problem_b_2(problem_b_2),
                                                                .solution_b(solution_b),
```

```
                                                    .problem_c_0(problem_c_0),
                                                    .problem_c_1(problem_c_1),
                                                    .problem_c_2(problem_c_2),
                                                    .problem_c_3(problem_c_3),
                                                    .solution_c(solution_c),
                                                    .problem_d_0(problem_d_0),
                                                    .problem_d_1(problem_d_1),
                                                    .problem_d_2(problem_d_2),
                                                    .problem_d_3(problem_d_3),
                                                    .solution_d(solution_d)
                                                );

problem2 #(.SIGNED_INPUT(1), .NONE_OF_THEM_ARE_ONE(3)) p2b(
                                                    .problem_a(problem_a),
                                                    .solution_a(solution_a2),
                                                    .problem_b_1(problem_b_1),
                                                    .problem_b_2(problem_b_2),
                                                    .solution_b(solution_b2),
                                                    .problem_c_0(problem_c_0),
                                                    .problem_c_1(problem_c_1),
                                                    .problem_c_2(problem_c_2),
                                                    .problem_c_3(problem_c_3),
                                                    .solution_c(solution_c2),
                                                    .problem_d_0(problem_d_0),
                                                    .problem_d_1(problem_d_1),
                                                    .problem_d_2(problem_d_2),
                                                    .problem_d_3(problem_d_3),
                                                    .solution_d(solution_d2)
                                                );

initial
begin
   $dumpfile("test.vcd");
        $dumpvars(0,problem2_tb);
        #10;

        op1 = 8'd16;
        op2a = 16'd5;
        op2b = 16'd127;
        i0 = 1;
        i1 = 1;
        i2 = 1;
        i3 = 1;
        #10;

        op1  = 8'b10100111;
        op2a = 16'd456;
        op2b = 16'd123;
        i0 = 1;
```

```
            i1 = 3;
            i2 = 3;
            i3 = 3;

            #10;
            i0 = 1;
            i1 = 1;
            i2 = 3;
            i3 = 3;

            #10;
            i0 = 1;
            i1 = 0;
            i2 = 1;
            i3 = 0;

            #10;
        end

endmodule
```

```
module problem3_tb;

	reg [15:0] data;
	wire even_parity;

	problem3 p3(
					.data_in(data),
					.even_parity_out(even_parity)
				);

	initial
	begin
	   $dumpfile("test.vcd");
		$dumpvars(0,problem3_tb);
		#10;

		data = 16'b11111;

		#10;

		data = 16'b1111;

		#10;
	end
endmodule
```