```
module gate_and (
  input wire x0,
  input wire x1,
  output wire y
);

  assign y = x0 & x1;

endmodule

// --------------------------------

module gate_or (
  input wire x0,
  input wire x1,
  output wire y
  );

  assign y = x0 | x1;

endmodule

// --------------------------------

module gate_xor (
  input wire x0,
  input wire x1,
  output wire y
  );

  assign y = x0 ^ x1;

endmodule

// --------------------------------

module gate_inv (
  input wire x0,
  output wire y
  );

  assign y = ~x0;

endmodule

// --------------------------------

module gate_oai3 (
  input wire x0,
  input wire x1,
  input wire x2,
  output wire y
  );

  assign y = ~((x0 | x1) & x2);

endmodule

// --------------------------------

module mux_2s_4d #(
  parameter N = 8
  )(
  input wire [N-1:0] x0,
  input wire [N-1:0] x1,
  input wire [N-1:0] x2,
  input wire [N-1:0] x3,
  input wire [1:0] s,
  output reg [N-1:0] y
```

```
    );

    always @(*) begin

          case (s)
            2'b00:   y = x0;
            2'b01:   y = x1;
            2'b10:   y = x2;
            default: y = x3;

          endcase

          // Otra solución
      // if (s==2'b00) begin
      //   y = x0;
      // end
      // else if (s==2'b01) begin
      //   y = x1;
      // end
      // else if (s==2'b10) begin
      //   y = x1;
      // end
      // else begin
      //   y = x3;
      // end

    end

  endmodule

  // ----------------------------
  // SIN SIGNO
  // ----------------------------

  module adder #(
    parameter N = 10
    )(
    input wire [N-1:0] x0,
    input wire [N-1:0] x1,
    output wire [N  :0] y
    );

    assign y = x0 + x1;

  endmodule


  // ----------------------------

  module adder_us #(
    parameter N = 10
    )(
    input  wire  [N-1:0] x0,
    input  wire  [N-1:0] x1,
    output wire  [N-1:0] y,
    output wire          cy
    );

    assign {cy,y} = x0 + x1;

  endmodule

  // ----------------------------
  // CON SIGNO
  // ----------------------------

  module adder_2c #(
    parameter N = 10
    )(
    input  wire  signed [N-1:0] x0,
    input   wire signed  [N-1:0] x1,
    output  wire signed [N-1:0] y
    );

    assign y = x0 + x1;

  endmodule

  // -----------------------------

  module mult_us #(
    parameter N = 10
    )(
    input  wire  [N-1:0] x0,
    input  wire  [N-1:0] x1,
    output wire  [2*N-1:0] y
    );

    assign y = x0 * x1;

  endmodule

  // -----------------------------

  module mult_2c #(
    parameter N = 10
    )(
    input   wire signed [N-1:0] x0,
    input   wire signed [N-1:0] x1,
    output  wire signed [2*N-1:0] y
    );

    assign y = x0 * x1;

  endmodule

  // -----------------------------

  module ffd #(
    parameter N = 10
    )(
    input  wire [N-1:0] d,
    input  wire         clk,
    input  wire         rst,
    output reg  [N-1:0] q
    );

    always @(posedge clk, posedge rst) begin

      if (rst) begin
        q <= {N{1'b0}};
      end
      else begin
        q <= d;
      end

    end

  endmodule

  module ffd_negedge #(
    parameter N = 10
  ) (
    input  wire [N-1:0] d,
    input  wire         clk,
    input  wire         rst,
    output reg  [N-1:0] q
    );
    always @(negedge clk, negedge rst) begin

      if (!rst) begin
        q <= {N{1'b0}};
      end
      else begin
        q <= d;
      end

    end

  endmodule

  // -----------------------------

  module contador #(
    parameter N = 10
    )(
    input  wire         clk,
    input  wire         rst,
    input  wire         enable,
    input  wire         load,
    input  wire [N-1:0] prox_cuenta,
    output reg  [N-1:0] cuenta
    );

    always @(posedge clk, posedge rst) begin

      if (rst) begin
        cuenta <= {N{1'b0}};
      end
      else begin

        if (enable) begin
          if (!load) begin
                  cuenta <= cuenta + {{(N-1){1'b0}},1'b1};
          end
          else begin
                  cuenta <= prox_cuenta;
          end
        end

      end

    end

  endmodule
```