module X (

input wire clk,

input wire reset,

output wire odd\_parity,

output wire even\_parity

);

wire [3:0] binary;

wire [3:0] bcd;

wire [3:0] digit;

BrunelIDNumberGenerator brunel\_id\_gen (

.clk(clk),

.reset(reset),

.digit(digit)

);

BinaryToBCDEncoder binary\_to\_bcd (

.binary(digit),

.bcd(bcd)

);

ParityGenerator parity\_gen (

.bcd(bcd),

.odd\_parity(odd\_parity),

.even\_parity(even\_parity)

);

endmodule

module BrunelIDNumberGenerator (

input wire clk,

input wire reset,

output reg [3:0] digit

);

reg [1:0] counter;

always @(posedge clk or posedge reset) begin

if (reset)

counter <= 0;

else if (counter == 2'h0)

counter <= 2'h1;

else if (counter == 2'h1)

counter <= 2'h2;

else

counter <= 2'h0;

end

always @(posedge clk) begin

if (counter == 2'h0)

digit <= 4'b1010; // Replace with your LS digit value

else if (counter == 2'h1)

digit <= 4'b0001; // Replace with your LS digit value

else if (counter == 2'h2)

digit <= 4'b0101; // Replace with your LS digit value

end

endmodule

module BinaryToBCDEncoder (

input wire [3:0] binary,

output reg [3:0] bcd

);

always @(\*) begin

case (binary)

4'b0000: bcd = 4'b0000;

4'b0001: bcd = 4'b0001;

4'b0010: bcd = 4'b0010;

4'b0011: bcd = 4'b0011;

4'b0100: bcd = 4'b0100;

4'b0101: bcd = 4'b0101;

4'b0110: bcd = 4'b0110;

4'b0111: bcd = 4'b0111;

4'b1000: bcd = 4'b1000;

4'b1001: bcd = 4'b1001;

default: bcd = 4'b0000; // Handle invalid input

endcase

end

endmodule

module ParityGenerator (

input wire [3:0] bcd,

output reg odd\_parity,

output reg even\_parity

);

reg [3:0] xor\_result;

always @(bcd) begin

xor\_result = bcd ^ 4'b1111;

odd\_parity = ^xor\_result;

even\_parity = ~^xor\_result;

end

endmodule