module X (

input wire clk,

input wire reset,

input wire enable,

output reg [3:0] digit,

output wire odd\_parity,

output wire even\_parity,

output reg [3:0] bcd

);

BrunelIDNumberGenerator brunel\_id\_gen (

.clk(clk),

.reset(reset),

.enable(enable),

.digit(digit\_wires)

);

BinaryToBCDEncoder binary\_to\_bcd (

.digit(digit),

.bcd(bcd\_wires) // Connecting the output bcd from BinaryToBCDEncoder module

);

ParityGenerator parity\_gen (

.bcd(bcd),

.odd\_parity(odd\_parity),

.even\_parity(even\_parity)

);

endmodule

module BrunelIDNumberGenerator (

input wire clk,

input wire reset,

input wire enable,

output reg [3:0] digit

);

reg [2:0] counter;

reg [3:0] currentDigit;

reg [2:0] pulseInterval;

always @(posedge clk or posedge reset) begin

if (reset)

counter <= 0;

else if (enable)

counter <= counter + 1;

end

always @(posedge clk) begin

if (reset)

pulseInterval <= 0;

else if (enable)

pulseInterval <= counter[2:0];

end

always @(posedge clk) begin

case (pulseInterval)

3'b010: currentDigit <= 4'b0010;

3'b101: currentDigit <= 4'b0100;

3'b110: currentDigit <= 4'b0111;

default: currentDigit <= 4'b0001;

endcase

end

always @(posedge clk) begin

if (reset)

digit <= 4'b0000;

else if (enable)

digit <= currentDigit;

end

endmodule//end the top level hirarchy

module BinaryToBCDEncoder (

input wire [3:0] digit,

output reg [3:0] bcd

);

always @(\*) begin

case (digit)

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//functrional level of Parity module to even na odd parity

//odd\_parity:check if the out\_bcd is odd

//If the input of bcd\_output is not equal to even as specified it is odd\_parity in 4'b0100 1 mean high 0 mean low

odd\_parity = (bcd != 4'b1111)&&(bcd != 4'b0011)&&(bcd != 4'b1100)&&(bcd != 4'b1001)&&(bcd != 4'b0110)&&(bcd != 4'b0000);

//even\_parity:check if the out\_bcd is even

//produce even\_parity if four or two on any bcd\_oput combinations are inputted(total input should be even numbers)

even\_parity = (bcd[0] & bcd[1] & bcd[2] & bcd[3]);//all four of bcd out is inputted

even\_parity = (bcd[0] & bcd[1] );//bcd\_out is input on only 0 and 1

even\_parity = (bcd[1] & bcd[2] );//bcd\_out is input on only 1 and 2

even\_parity = (bcd[2] & bcd[3] );//bcd\_out is input on only 2 and 3

even\_parity = (bcd[3] & bcd[1] );//bcd\_out is input on only 3 and 1

even\_parity = (bcd[3] & bcd[0] );//bcd\_out is input on only 3 and 0

end//end of functional level

endmodule