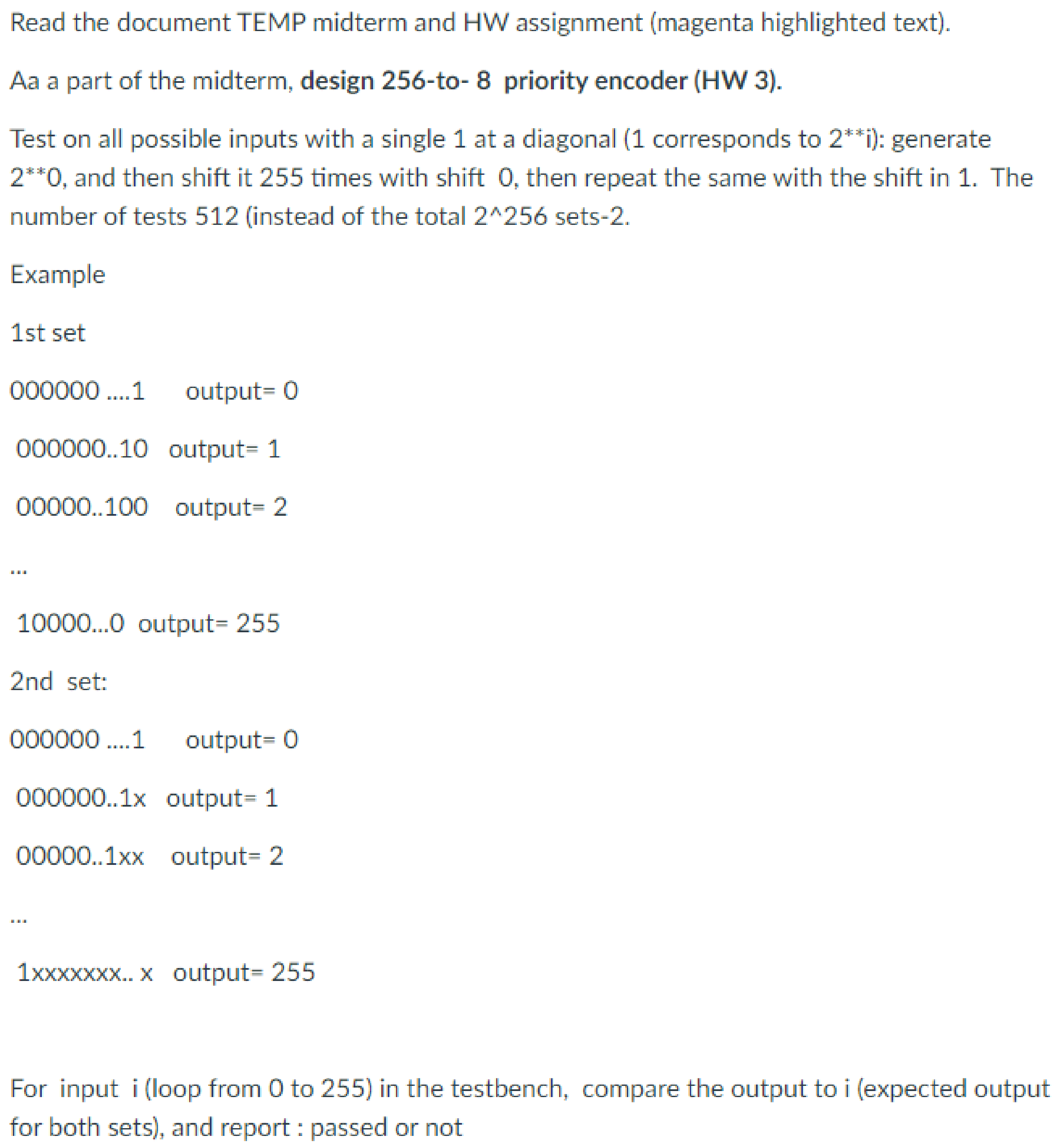
Parajes, Katherine Anne
CPE 409 - 1002
Homework 3



Read the document TEMP midterm and HW assignment (magenta highlighted text).

Aa a part of the midterm, **design 256-to- 8 priority encoder (HW 3).**

Test on all possible inputs with a single 1 at a diagonal (1 corresponds to 2**i): generate 2**0, and then shift it 255 times with shift 0, then repeat the same with the shift in 1. The number of tests 512 (instead of the total 2^256 sets-2.

Example

1st set

000000 ....1 output= 0

000000..10 output= 1

00000..100 output= 2

...

10000...0 output= 255

2nd set:

000000 ....1 output= 0

000000..1x output= 1

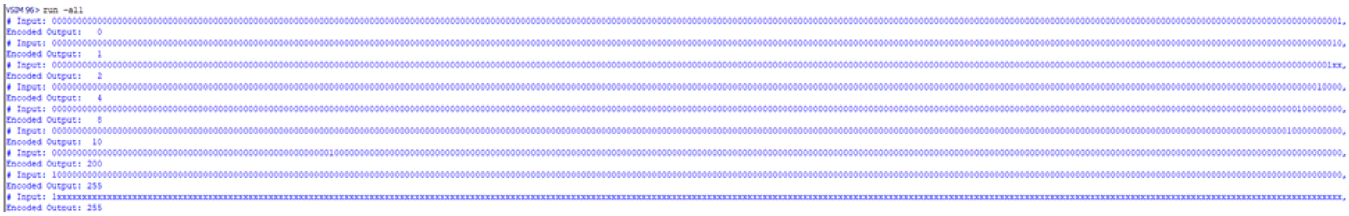
00000..1xx output= 2

...

1xxxxxxx.. x output= 255

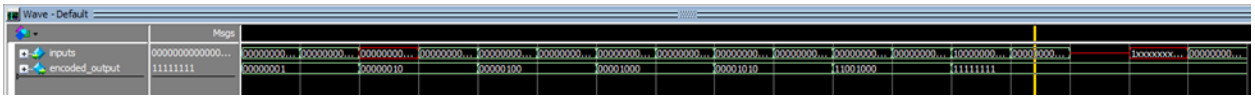
For input i (loop from 0 to 255) in the testbench, compare the output to i (expected output for both sets), and report : passed or not

Testbench results:

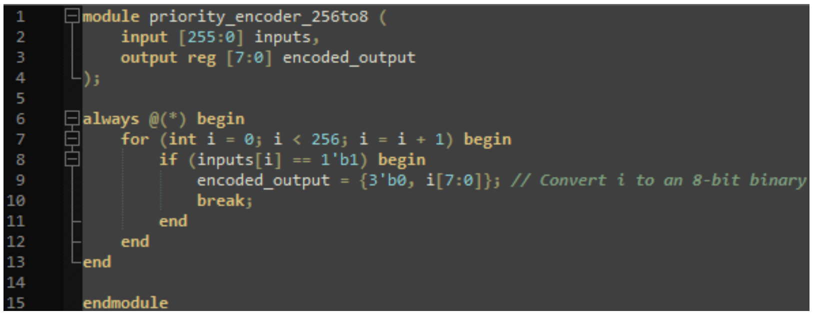


```
VSIM 96> run -all
# Input: 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001,
Encoded Output:   0
# Input: 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010,
Encoded Output:   1
# Input: 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001xx,
Encoded Output:   2
# Input: 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000,
Encoded Output:   4
# Input: 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000100000000,
Encoded Output:   8
# Input: 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000,
Encoded Output:  10
# Input: 0000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000,
Encoded Output: 200
# Input: 1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000,
Encoded Output: 255
# Input: 1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,
Encoded Output: 255
```

Waveforms:



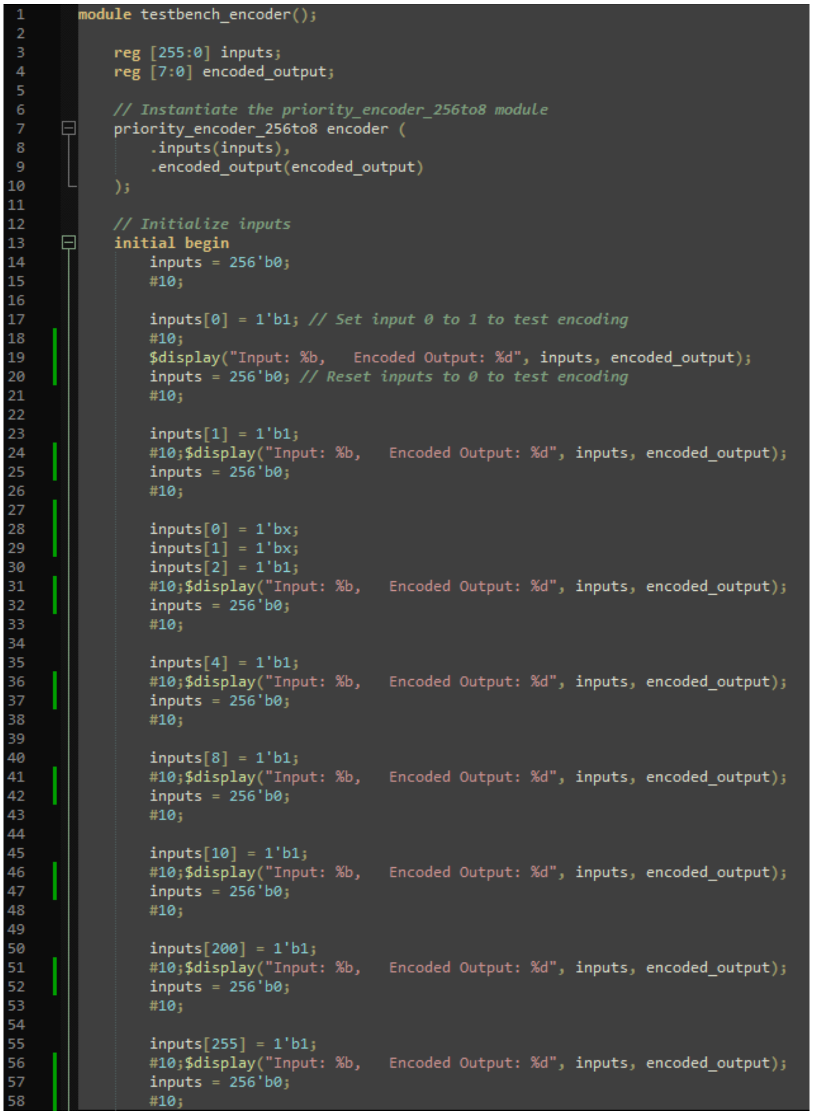
Code:



```
module priority_encoder_256to8 (
    input [255:0] inputs,
    output reg [7:0] encoded_output
);

always @(*) begin
    for (int i = 0; i < 256; i = i + 1) begin
        if (inputs[i] == 1'b1) begin
            encoded_output = {3'b0, i[7:0]}; // Convert i to an 8-bit binary
            break;
        end
    end
end

endmodule
```



```
module testbench_encoder();

    reg [255:0] inputs;
    reg [7:0] encoded_output;

    // Instantiate the priority_encoder_256to8 module
    priority_encoder_256to8 encoder (
        .inputs(inputs),
        .encoded_output(encoded_output)
    );

    // Initialize inputs
    initial begin
        inputs = 256'b0;
        #10;

        inputs[0] = 1'b1; // Set input 0 to 1 to test encoding
        #10;
        $display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0; // Reset inputs to 0 to test encoding
        #10;

        inputs[1] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;

        inputs[0] = 1'bx;
        inputs[1] = 1'bx;
        inputs[2] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;

        inputs[4] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;

        inputs[8] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;

        inputs[10] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;

        inputs[200] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;

        inputs[255] = 1'b1;
        #10;$display("Input: %b,   Encoded Output: %d", inputs, encoded_output);
        inputs = 256'b0;
        #10;
```