1 Computer Design (SWM)

Below is a functionally and syntactically correct mysterious module written in SystemVerilog.

typedef enum { opNone, opIn, opOut } operationT;

module mystery
    # (parameter depth, parameter width)
    (
        input clk,
        input rst,
        input operationT op,
        input logic [width-1:0] dataIn,
        output logic [width-1:0] dataOut,
        output logic empty,
        output logic full,
        output logic error);

    logic [width-1:0] mem[depth-1:0];
    reg [$clog2(depth-1)+1:0] head;
    // where $clog2(x) = ceiling(log_base_2(x))

    always_comb
        begin
            full = head>=depth;
            empty = head==0;
            error = ((op==opIn) && full) || ((op==opOut) && empty);
            dataOut = empty ? -1 : mem[head-1];
        end

    always @(posedge clk)
        if(rst)
            head <= 0;
        else
            if(!error)
                case(op)
                    opIn: begin head <= head+1; mem[head] <= dataIn; end
                    opOut: head <= head-1;
                endcase // case (op)
    endmodule

(a) What is the function of the mystery module? Include in your answer the behaviour when the module is full (full==1) or empty (empty==1). What does dataOut output? What does input op do? [8 marks]

(b) What is production test and how does it differ from functional test? [2 marks]

(c) What are the key challenges in functionally testing the mystery module? [5 marks]

(d) What are the challenges in undertaking a production test of the mystery module and how do these challenges compare with those for functional test? [5 marks]