FIFO after fixing the following Bugs:

1. bug detected : Reset should makes only these seq outputs(overflow, under flow, wr_ack).

2. bug detected : output (underflow), It must be sequential not combinational output.

3. bug detected : output(almostfull), It must = 1 ,if count = FIFO_DEPTH - 1 , not FIFO_DEPTH – 2.

4. bug detected : the third always block should contain case of (1,1) to ensure that note """If a read and write enables were high and the FIFO was empty, only writing will take place and vice verse if the FIFO was full."""

```
module FIFO(FIFO_interface.DUT fifo_if);

    logic [fifo_if.FIFO_WIDTH-1:0] data_in;
    logic clk, rst_n, wr_en, rd_en;
    logic [fifo_if.FIFO_WIDTH-1:0] data_out;
    logic wr_ack, overflow, full, empty, almostfull, almostempty, underflow;

    assign clk         = fifo_if.clk;
    assign rst_n       = fifo_if.rst_n;
    assign wr_en       = fifo_if.wr_en;
    assign rd_en       = fifo_if.rd_en;
    assign data_in     = fifo_if.data_in;
    assign data_out    = fifo_if.data_out;
    assign wr_ack      = fifo_if.wr_ack;
    assign overflow    = fifo_if.overflow;
    assign full        = fifo_if.full;
    assign empty       = fifo_if.empty;
    assign almostfull  = fifo_if.almostfull;
    assign almostempty = fifo_if.almostempty;
    assign underflow   = fifo_if.underflow;

    reg [fifo_if.FIFO_WIDTH-1    :0] mem [fifo_if.FIFO_DEPTH-1:0];
    reg [fifo_if.max_fifo_addr-1 :0] wr_ptr, rd_ptr;
    reg [fifo_if.max_fifo_addr   :0] count;

    always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin //////write operation///////
        if (!fifo_if.rst_n) begin
            wr_ptr           <= 0;
            fifo_if.wr_ack   <= 0;
            fifo_if.overflow <= 0;
        end

        else if (fifo_if.wr_en && count < fifo_if.FIFO_DEPTH) begin
            mem[wr_ptr]      <= fifo_if.data_in;
            fifo_if.wr_ack   <= 1;
            wr_ptr           <= wr_ptr + 1;
            fifo_if.overflow <= 0;
        end

        else begin
            fifo_if.wr_ack <= 0;

            if (fifo_if.wr_en && fifo_if.full)
                fifo_if.overflow <= 1;
            else
                fifo_if.overflow <= 0;
            end
    end

    always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin ///////read operation///////
        if (!fifo_if.rst_n) begin
            rd_ptr           <= 0;
            fifo_if.underflow <= 0;
            //fifo_if.data_out  <= 0;
        end
```

```
always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin ///////read operation///////
    if (!fifo_if.rst_n) begin
        rd_ptr            <= 0;
        fifo_if.underflow <= 0;
        //fifo_if.data_out  <= 0;
    end

    else if (fifo_if.rd_en && count != 0) begin
        fifo_if.data_out  <= mem[rd_ptr];
        rd_ptr            <= rd_ptr + 1;
        fifo_if.underflow <= 0;
    end

    else begin

        if (fifo_if.rd_en && fifo_if.empty)
            fifo_if.underflow <= 1;
        else
            fifo_if.underflow <= 0;

    end
end

always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        count <= 0;
    end
    else begin
        if      ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b10) && !fifo_if.full)
                count <= count + 1;

        else if ( ({fifo_if.wr_en, fifo_if.rd_en} == 2'b01) && !fifo_if.empty)
                count <= count - 1;

        else if   ({fifo_if.wr_en, fifo_if.rd_en} == 2'b11) begin

            if (fifo_if.full)
                count <= count - 1;
            else if (fifo_if.empty)
                count <= count + 1;
            //else
                //count <= count;

        end
    end
end

assign fifo_if.full        = (count == fifo_if.FIFO_DEPTH)   ? 1 : 0;
assign fifo_if.empty       = (count == 0)                    ? 1 : 0;
assign fifo_if.almostfull  = (count == fifo_if.FIFO_DEPTH-1) ? 1 : 0;
assign fifo_if.almostempty = (count == 1)                    ? 1 : 0;



`ifdef SIM
```