

```

module write_logic#(parameter depth = 7, width = 8)(input
clk_in,insert,reset,flush,input [depth : 0]r2wsync_ff2,input[31:0]
data_in, output reg [depth-1 : 0]write_addr, output reg [depth : 0]wptr,
output reg write_enable,full,output reg[31:0] temp_data_in);
//parameter[1:0] idle=2'b00, s0=2'b01,s1=2'b11;
//reg[1:0] current_state, next_state;
parameter[2:0] idle=3'b001, s0=3'b010,s1=3'b100;
reg[2:0] current_state, next_state;
reg [31:0] temp4;
always@(posedge clk_in, negedge reset)
begin
    if(!reset)
        current_state<=idle;
    else if(flush)
        current_state<=idle;
    else
        current_state<=next_state;
end
always@(*)
begin

case(current_state)
    idle:
        if(insert)
            next_state=s0;
        else
            next_state=idle;
    s0:
        if(insert)
            next_state=s0;
        else
            next_state=s1;
    s1:
        if(insert)
            next_state=s0;
        else
            next_state=s1;
        default:    next_state=idle;
endcase
end
always@(posedge clk_in, negedge reset)
begin
    if(!reset)
    begin
        full<=0;
        write_enable <= 0;
        write_addr <= 0;
        wptr <=0;
    end
    else if(flush)
    begin
        full<=0;
        write_enable <= 0;
        write_addr <= 0;
    end
end

```

```

        wptr <=0;
    end
    else
    case(current_state)
        idle:
        begin
            full<=0;
            write_enable <= 0;
            write_addr <= 0;
            wptr <=0;

        end
        s0:
        begin
            if(wptr[depth-1:0] == r2wsync_ff2[depth-1:0] &&
wptr[depth] != r2wsync_ff2[depth])
            begin
                full<=1'b1;
                write_enable <= 1'b0;
                write_addr <= write_addr;
                wptr <= wptr;

            end
            else
            begin
                full<=1'b0;
                write_enable <= 1'b1;
                write_addr <= wptr[width-2:0];
                if(wptr == 8'b11111111)
                    wptr <= 8'h0;
                else begin
                    wptr <= wptr+1;
                end

            end

        end
        end
        s1:
        begin
            if(!(wptr[depth-1:0] == r2wsync_ff2[depth-1:0] &&
wptr[depth] != r2wsync_ff2[depth]))
            begin
                full<=0;
                write_enable <= 0;
                write_addr <= write_addr;
                wptr <= wptr;

            end
            else
            begin
                full<=full;
                write_enable <= 0;
                write_addr <= write_addr;
                wptr <= wptr;

            end

        end
    endcase
end

```

```
end
always @ (posedge clk_in or negedge reset)
begin
    if(!reset)
        temp_data_in<=32'h0;
    else
        begin
            temp4<=data_in;
            temp_data_in<=temp4;
        end
    end
end
endmodule
```