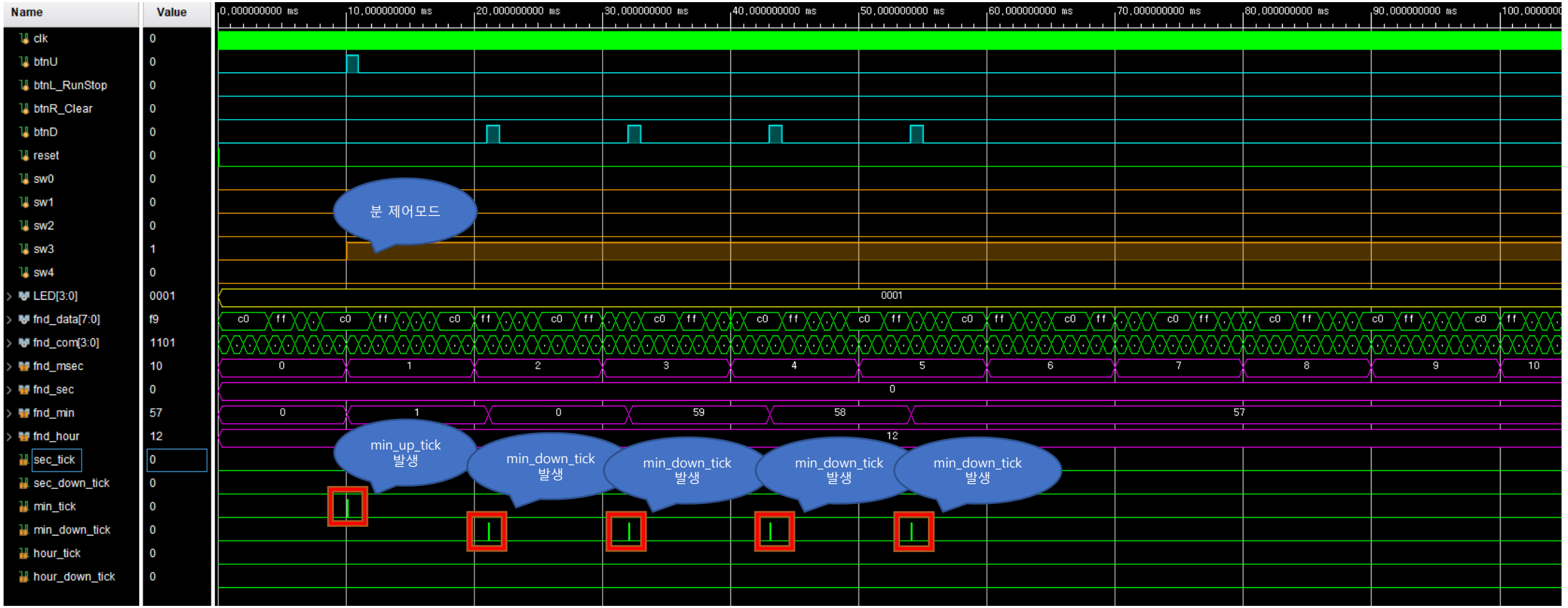


	btnU	btnD
sw2	sec 증가	sec 감소
sw3	min 증가	min 감소
sw4	hour 증가	hour 감소

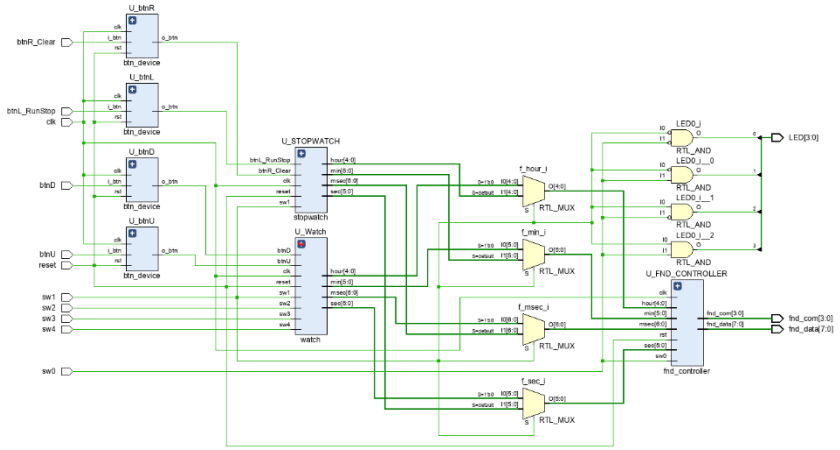
SIMULATION

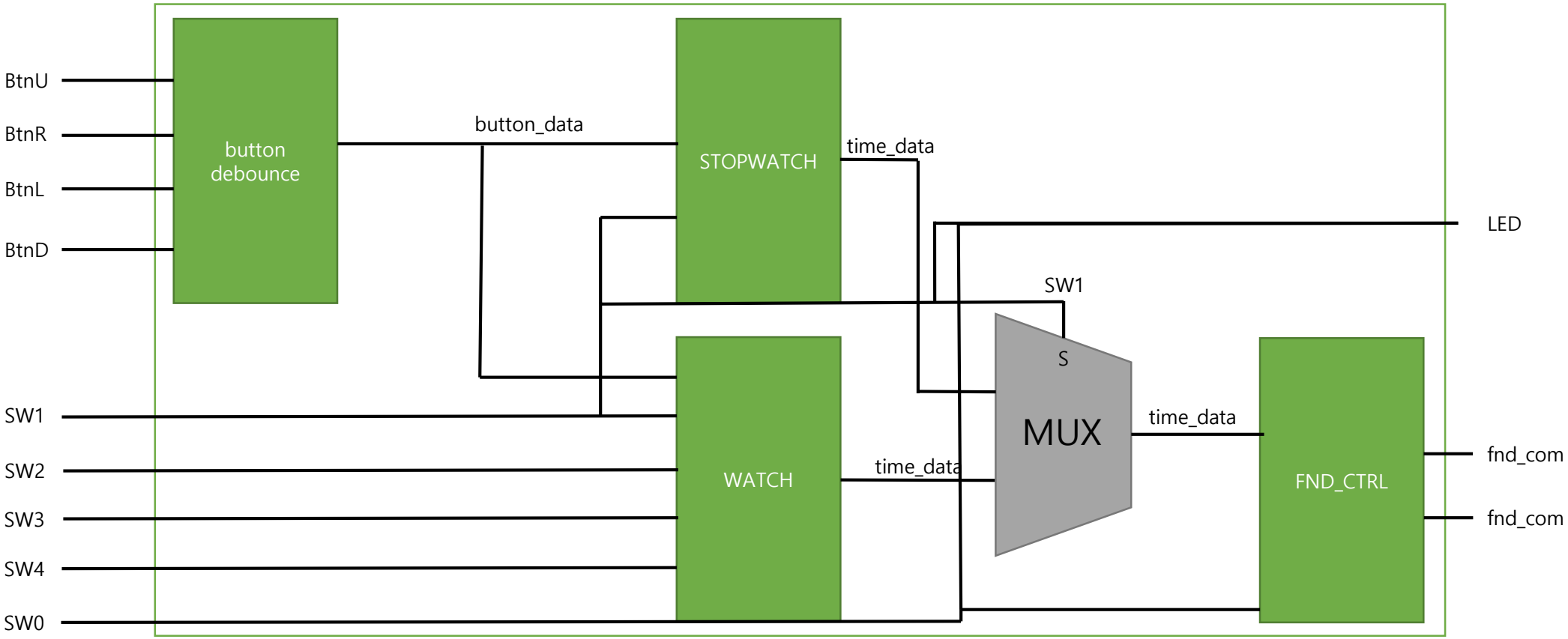
input 정리

	data	
	0	1
sw0	msec / sec fnd 출력	min / hour fnd 출력
sw1	시계 모드	스톱워치 모드

	btnU	btnD
sw2	sec 증가	sec 감소
sw3	min 증가	min 감소
sw4	hour 증가	hour 감소

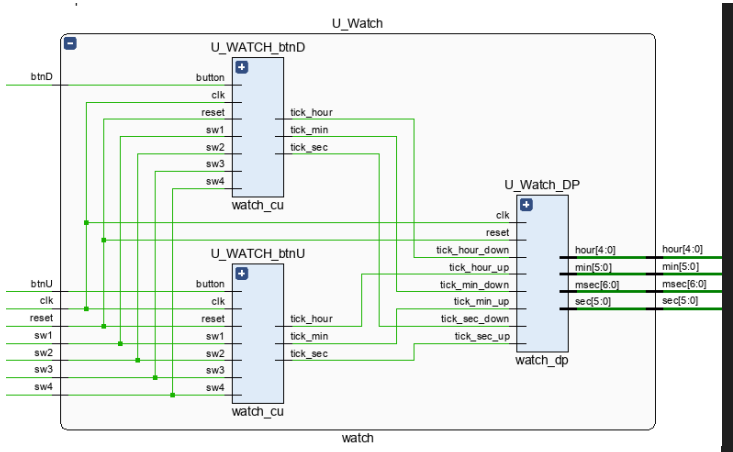
TOP MODULE





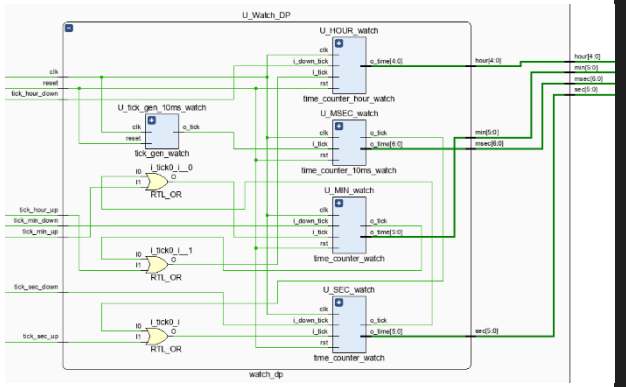

```
assign f_msec = (sw1 == 0) ? w_msec : s_msec;
assign f_sec = (sw1 == 0) ? w_sec : s_sec;
assign f_min = (sw1 == 0) ? w_min : s_min;
assign f_hour = (sw1 == 0) ? w_hour : s_hour;

assign LED[0] = (~sw1 & ~sw0);
assign LED[1] = (~sw1 & sw0);
assign LED[2] = (sw1 & ~sw0);
assign LED[3] = (sw1 & sw0);

btn_device U_btnR(
    .clk(clk),
    .rst(reset),
    .i_btn(btnR_Clear),
    .o_btn(w_btnR)
);

btn_device U_btnL(
    .clk(clk),
    .rst(reset),
    .i_btn(btnL_RunStop),
    .o_btn(w_btnL)
);

btn_device U_btnU(
    .clk(clk),
    .rst(reset),
    .i_btn(btnU),
    .o_btn(w_btnU)
);

btn_device U_btnD(
    .clk(clk),
    .rst(reset),
    .i_btn(btnD),
    .o_btn(w_btnD)
);

stopwatch U_STOPWATCH (
    .clk(clk),
    .reset(reset),
    .sw1(sw1),
    .btnR_Clear(w_btnR),
    .btnL_RunStop(w_btnL),
    .msec(s_msec),
    .sec(s_sec),
    .min(s_min),
    .hour(s_hour)
);

watch U_Watch(
    .clk(clk),
    .reset(reset),
    .btnU(w_btnU),
    .sw1(sw1),//sw1이 0인 경우 시계 모드, 시계모드
    .sw2(sw2),//초바꾸기
    .sw3(sw3),//분바꾸기
    .sw4(sw4),//시바꾸기
    .btnD(w_btnD),
    .msec(w_msec),
    .sec(w_sec),
    .min(w_min),
    .hour(w_hour)
);

fnd_controller U_FND_CONTROLLER (
    .clk(clk),
    .rst(reset),
    .sw0(sw0),
    .msec(f_msec),
    .sec(f_sec),
    .min(f_min),
    .hour(f_hour),
    .fnd_data(fnd_data),
    .fnd_com(fnd_com)
);
```

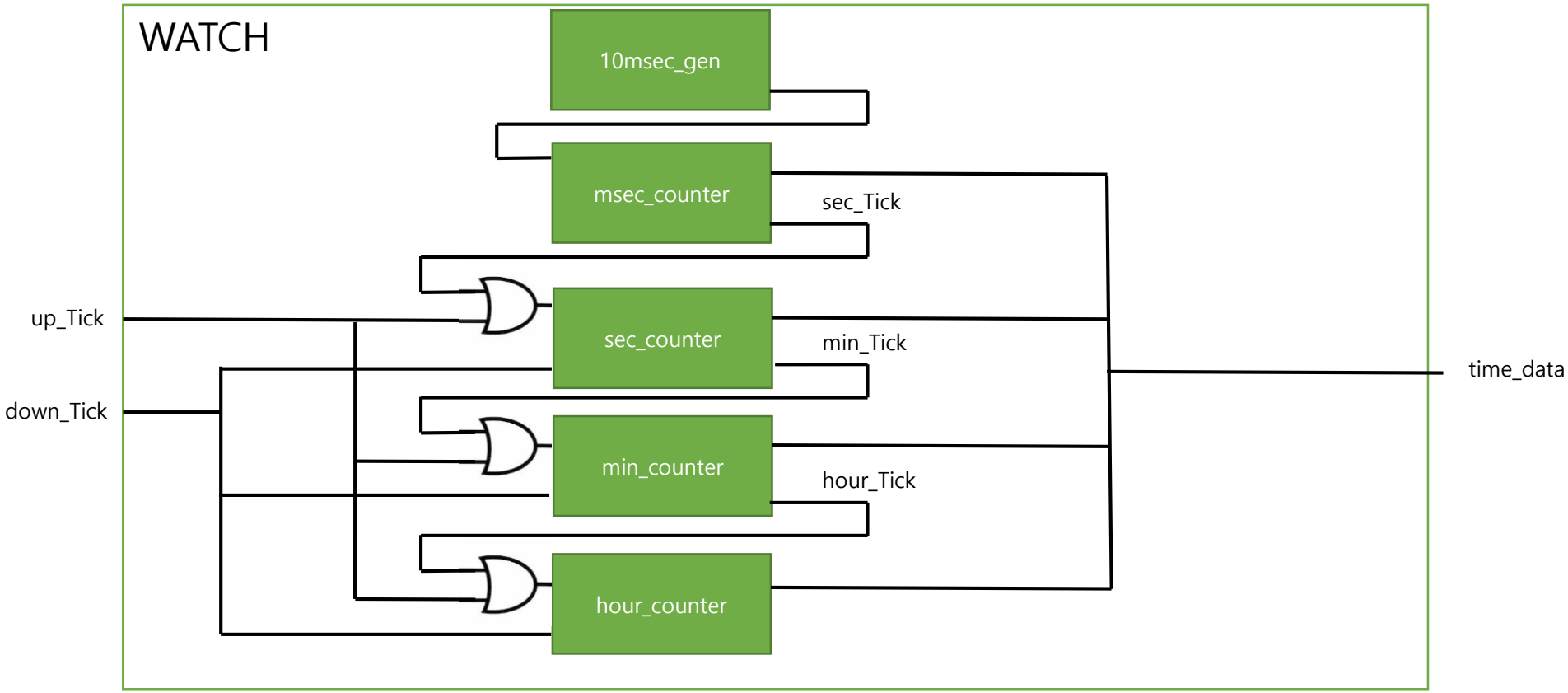
WATCH MODULE




```
watch_cu U_WATCH_btnU (
    .clk(clk),
    .reset(reset),
    .button(btnU),
    .sw1 (sw1),
    .sw2 (sw2),
    .sw3 (sw3),
    .sw4 (sw4),
    .tick_sec(w_tick_sec_up),
    .tick_min(w_tick_min_up),
    .tick_hour(w_tick_hour_up)
);

watch_cu U_WATCH_btnD (
    .clk(clk),
    .reset(reset),
    .button(btnD),
    .sw1 (sw1),
    .sw2 (sw2),
    .sw3 (sw3),
    .sw4 (sw4),
    .tick_sec(w_tick_sec_down),
    .tick_min(w_tick_min_down),
    .tick_hour(w_tick_hour_down)
);

watch_dp U_Watch_DP (
    .clk  (clk),
    .reset(reset),
    .tick_sec_up(w_tick_sec_up),
    .tick_min_up(w_tick_min_up),
    .tick_hour_up(w_tick_hour_up),
    .tick_sec_down(w_tick_sec_down),
    .tick_min_down(w_tick_min_down),
    .tick_hour_down(w_tick_hour_down),
    .msec (msec),
    .sec  (sec),
    .min  (min),
    .hour (hour)
);
```

WATCH_DP MODULE






```
tick_gen_watch U_tick_gen_10ms_watch( // 10ms 생성
    .clk(clk),
    .reset(reset),
    .o_tick(w_msec_tick)
);

time_counter_10ms_watch #(.TICK_COUNT(100)) U_MSEC_watch (
    .clk(clk),
    .rst(reset),
    .i_tick(w_msec_tick),
    .o_time(msec),
    .o_tick(w_sec_tick)
);

time_counter_watch #(.TICK_COUNT(60)) U_SEC_watch (
    .clk(clk),
    .rst(reset),
    .i_tick(w_sec_tick | tick_sec_up),
    .i_down_tick(tick_sec_down),
    .o_time(sec),
    .o_tick(w_min_tick)
);

time_counter_watch #(.TICK_COUNT(60)) U_MIN_watch (
    .clk(clk),
    .rst(reset),
    .i_tick(w_min_tick | tick_min_up),
    .i_down_tick(tick_min_down),
    .o_time(min),
    .o_tick(w_hour_tick)
);

time_counter_hour_watch #(.TICK_COUNT(24)) U_HOUR_watch (
    .clk(clk),
    .rst(reset),
    .i_tick(w_hour_tick | tick_hour_up),
    .i_down_tick(tick_hour_down),
    .o_time(hour),
    .o_tick()
);
module time_counter_watch #(//초,분
    parameter TICK_COUNT = 100
) (
    input                           clk,
    input                           rst,
    input                           i_tick,
    input                           i_down_tick,
    output [$clog2(TICK_COUNT)-1:0] o_time,
    output                          o_tick
);

    reg [$clog2(TICK_COUNT)-1:0] count_reg, count_next;
    reg o_tick_reg, o_tick_next;

    assign o_time = count_reg;
    assign o_tick = o_tick_reg;

    // state register
    always @(posedge clk, posedge rst) begin
        if(rst) begin
            count_reg <= 0;// nonblock => f/f 사용할때 일반적으로
            o_tick_reg <= 1'b0;
        end else begin
            count_reg <= count_next;
            o_tick_reg <= o_tick_next;
        end
    end

    // next state
    always @(*) begin // Combinational logic으로 구현
        count_next = count_reg;
        o_tick_next = 1'b0;

        if(i_tick == 1'b1) begin
            if (count_reg == (TICK_COUNT - 1)) begin
                count_next = 0;
                o_tick_next = 1'b1;
            end else begin
                count_next = count_reg + 1;
                o_tick_next = 1'b0;
            end
        end
        if(i_down_tick == 1'b1) begin
            count_next = (count_reg == 0) ? 59 : count_reg - 1;
        end
    end
endmodule
```