Project

# FPGA 기반
# Watch/Stopwatch 설계 구현

황석현

# 목차

# 1. Introduction

## - 목적

FPGA 보드를 이용한 시계 및 스톱워치 구현

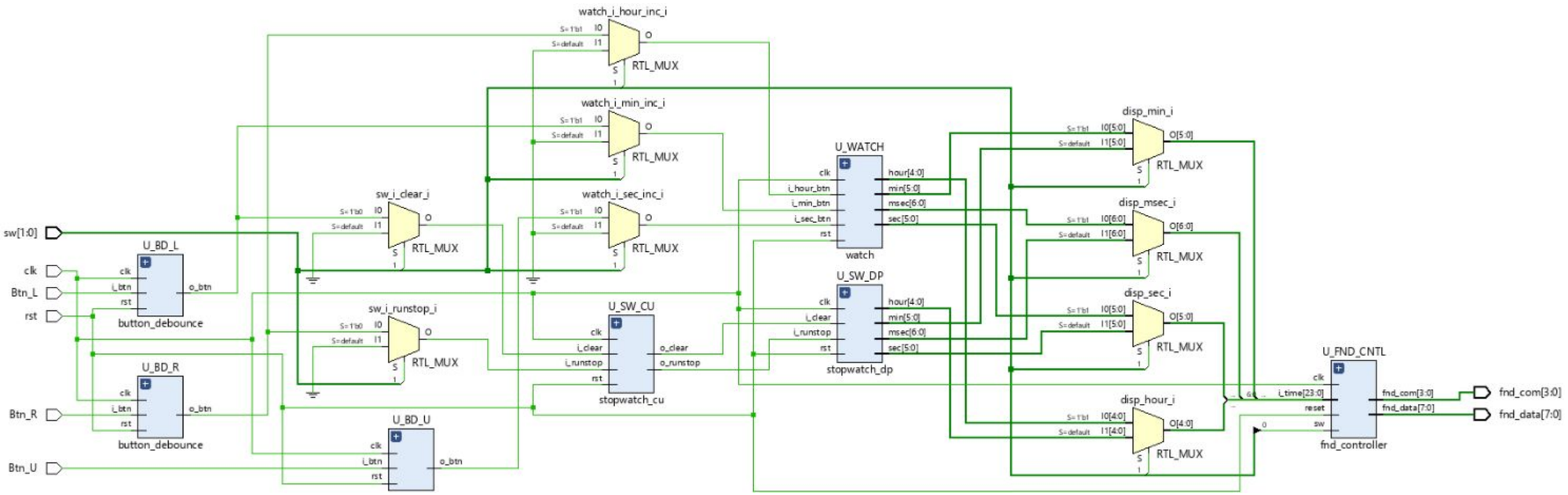
## - 주요 기능

- 버튼 입력 : stopwatch 시작과 정지
  watch 시간 변경
- 버튼 디바운싱 : 버튼 노이즈 처리
- 스위치 입력 : 시-분, 초-밀리초 표시 전환
- 출력 장치 : Basys 3 보드 7-Segment Display

## - 작업 환경


VIVADO
HL*x* Editions

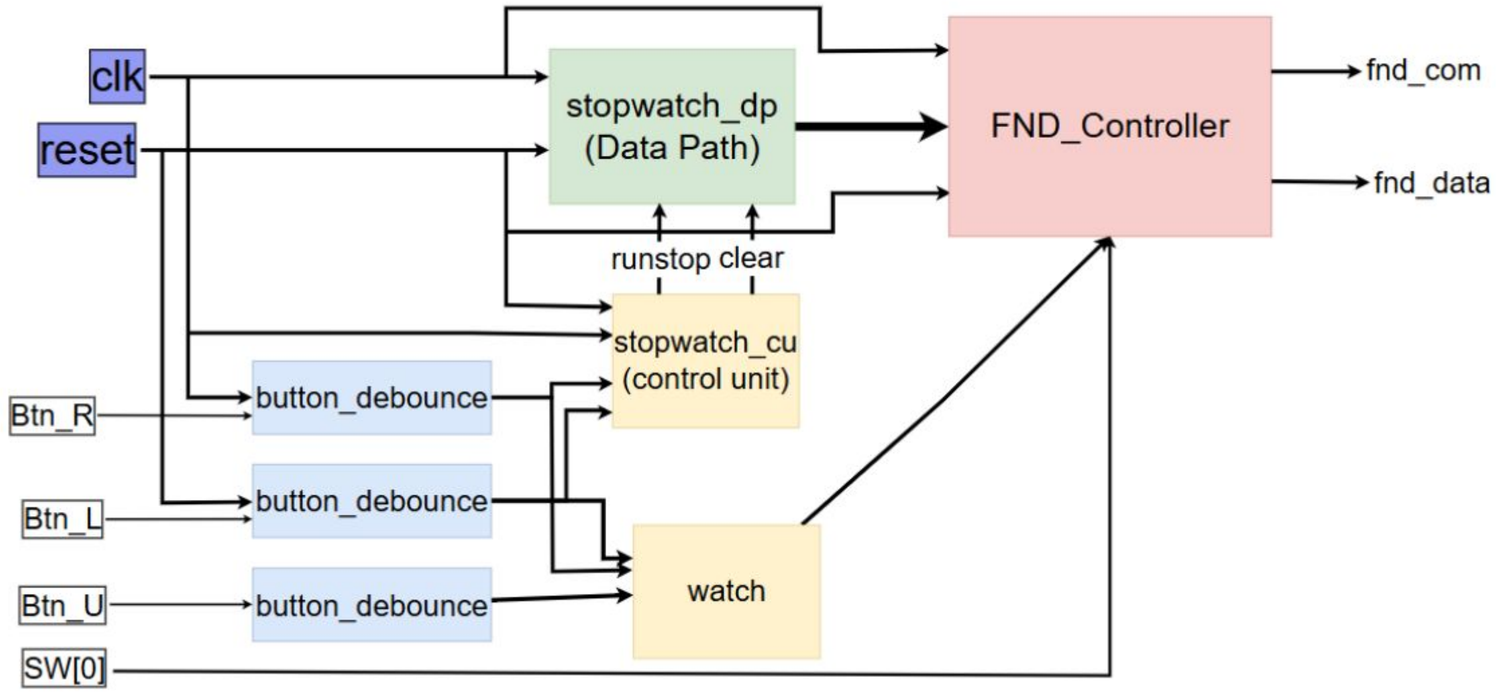
# 2. Schematic




- 물리 버튼 입력 Debounce 처리 → stopwatch_cu에 전달 → 입력에 따라 Run/Stop/Clear 상태 stopwatch_dp에 전달
  → stopwatch_dp가 시간(msec→sec→min→hour)을 계산하여 FND Controller로 전달 → 7-segment에 표시

- watch 모듈
  : 하위 카운터 오버플로우 시 상위 카운터로 tick 전달 → 최종 시간 값이 FND Controller를 거쳐 시·분·초·밀리초로 표시

# 1) stopwatch.v




Input : clk, rst, Btn_L, Btn_R, Btn_U, sw

Output : fnd_com[3:0], fnd_data[7:0]

- 기능 : 전체 서브모듈(button_debounce, stopwatch_cu, stopwatch_dp, FND Controller) 연결

# 2) button_debounce.v

```
always @(posedge clk_reg, posedge rst) begin
    if (rst) begin
        q_reg <= 0;
    end else begin
        q_reg <= q_next;
    end
end

always @(*) begin
    q_next = {i_btn, q_reg[7:1]};
end

assign debounce = &q_reg;

always @(posedge clk, posedge rst) begin
    if (rst) begin
        edge_reg <= 1'b0;
    end else begin
        edge_reg <= debounce;
    end
end

assign o_btn = ~edge_reg & debounce;
```

- 구성

1. Shift register

   posedge clk_reg에서 {i_btn, q_reg[7:1]}로 갱신

2. Debounce detect

   8bit 모두 1일 때(debounce = &q_next) 신호 안정화 판단

3. Edge detector

   이전 엣지 상태와 debounce 비교 후 상승 엣지 감지 시 o_btn 출력

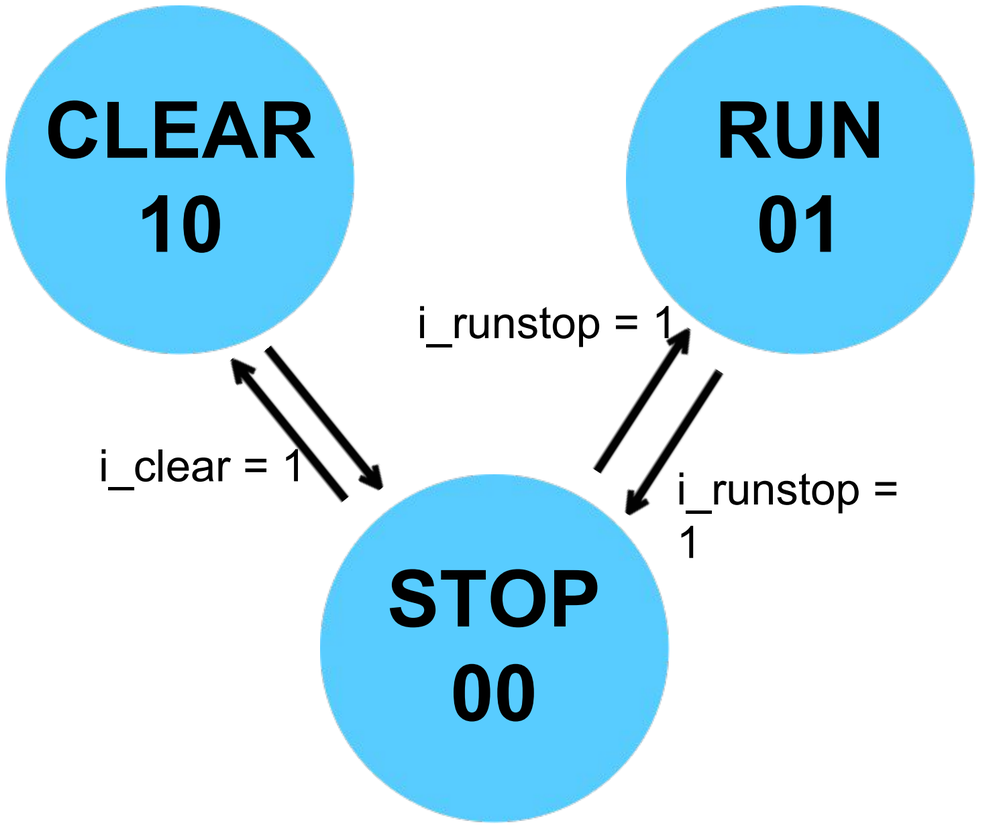
- 동작

8bit 시프트 레지스터에 연속 샘플 저장

8bit 모두 1로 유지되면 “안정적으로 눌림”으로 판단

안정화된 버튼 입력에서 상승 엣지를 감지하여 출력

# 3) stopwatch_cu.v




- 구성

상태: STOP(2'b00), RUN(2'b01), CLEAR(2'b10)

- 동작

외부 버튼 이벤트에 따라 상태 전환

각 상태에서 동작 신호를 출력하여 카운터 및 디스플레이 제어

STOP : o_runstop=0, o_clear=0

i_runstop=1 입력 시 → RUN 상태 전이

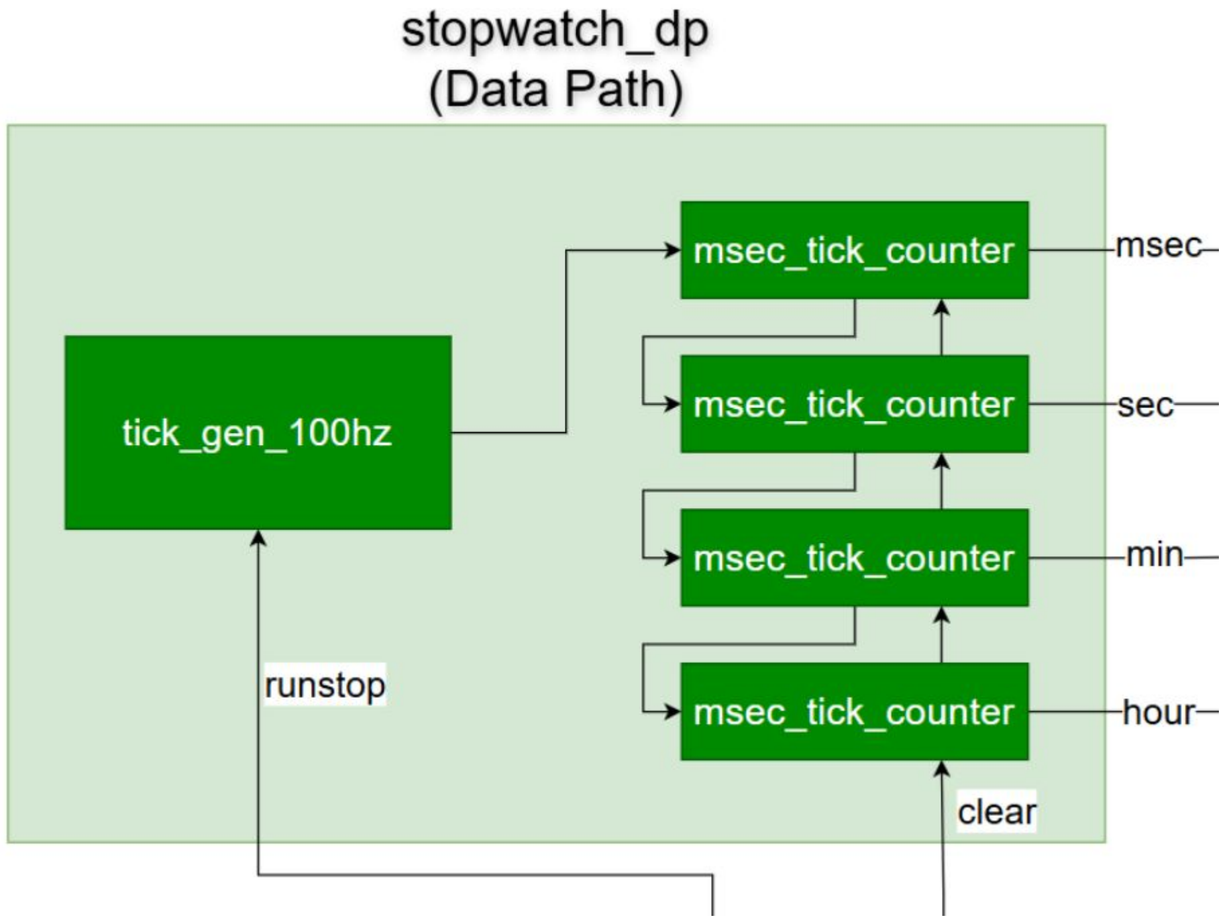
i_clear=1 입력 시 → CLEAR 상태 전이

RUN : o_runstop=1

i_runstop=1 입력 시 → STOP 상태 전이

CLEAR : o_clear=1 (1클럭 펄스 발생) 후 → STOP 상태 전이

# 4) stopwatch_dp.v




- 구성

tick_gen_100hz: 100MHz → 100Hz 클럭 분주

time_counter 모듈 계층적 연결

: 밀리초 → 초 → 분 → 시간 카운터 연결

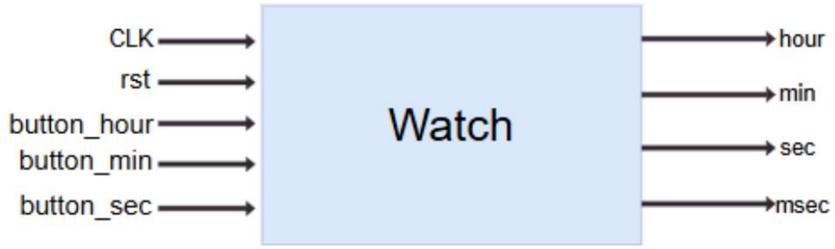
- 동작

i_runstop=1 상태에서만 카운팅 진행

하위 카운터 오버플로우 시 상위 카운터로 tick 전달

i_clear=1 입력 시 모든 카운터 즉시 초기화

# 5) watch.v




```
always @(posedge clk_100hz or posedge rst) begin
    if (rst) begin
        msec <= 0; sec <= 0; min <= 0; hour <= 12;
    end else begin
        if (msec == 99) begin
            msec <= 0;
            if (!sec_pulse) begin
                if (sec == 59) sec <= 0; else sec <= sec + 1;
                // 분, 시 증가도 같은 방식으로 처리
            end
        end else begin
            msec <= msec + 1;
        end
        if (sec_pulse) sec <= (sec == 59) ? 0 : sec + 1;
        // 분, 시 버튼 펄스 처리도 동일
    end
end
```

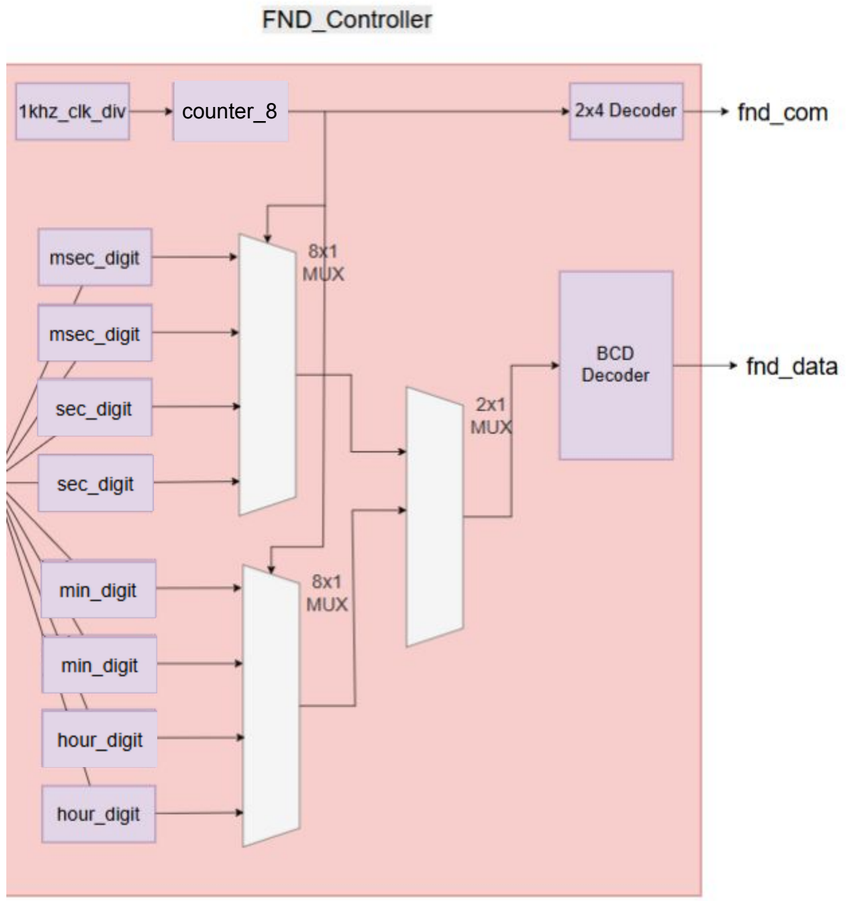
- 구성

  밀리초 → 초 → 분 → 시 자동 카운트
  버튼 입력 시 각 시간 단위 수동 증가

- 동작

  0.01초 마다 밀리초 증가, 99→0일 때 초 증가
  초 59→0일 때 분 증가, 분 59→0일 때 시 증가
  버튼 누르면 해당 시간 1씩 증가
  리셋 시 초기 시각(12시)으로 초기화

# 6) fnd_controller.v




- 구성

counter_8 : 0~3 카운터 (자리 선택)

decoder_2x4 : 자리 선택 → fnd_com

digit_splitter : 10진수 분리(/, % 연산)

mux_8x1, mux_2x1 : 시간 자리/모드 선택

bcd_decoder : BCD → 7-seg

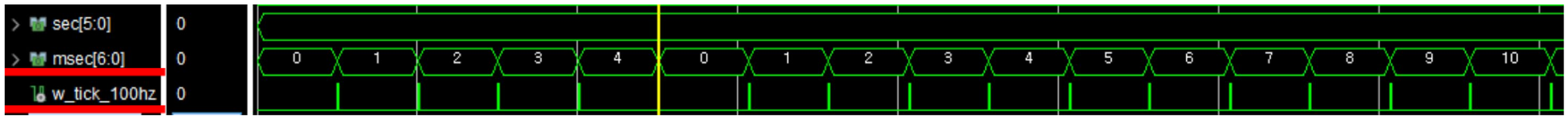
- 동작

sw 값에 따라 (밀리초, 초) 또는 (분, 시간) 값 선택 출력

10진수 숫자를 BCD로 분리하여 7-세그먼트 신호로 변환

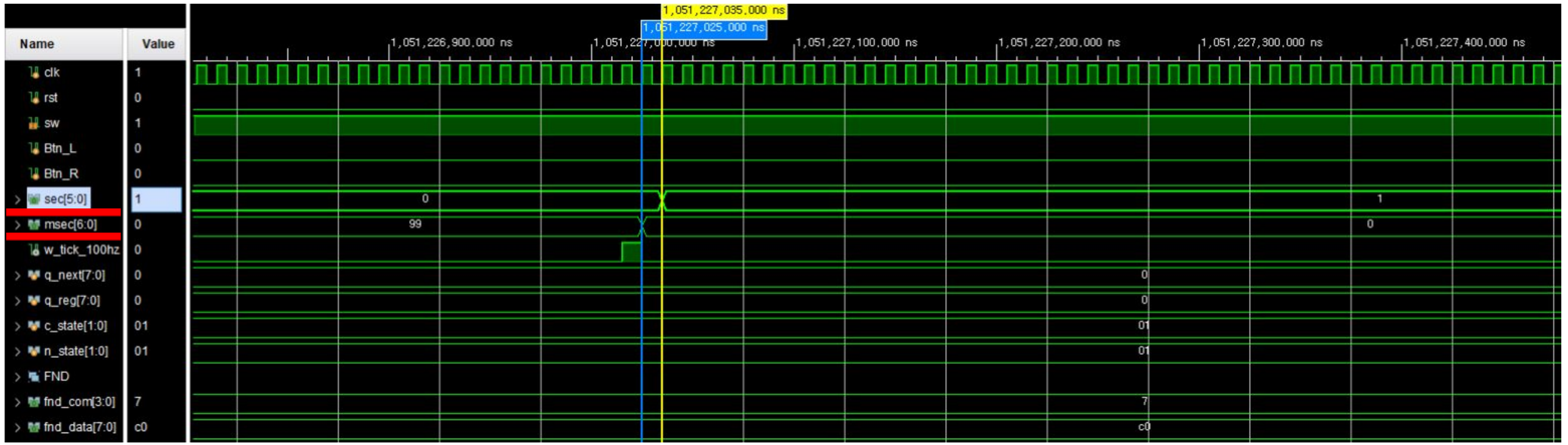
네 숫자 가운데 깜빡이는 점 구현

# 3. 시뮬레이션 (1)



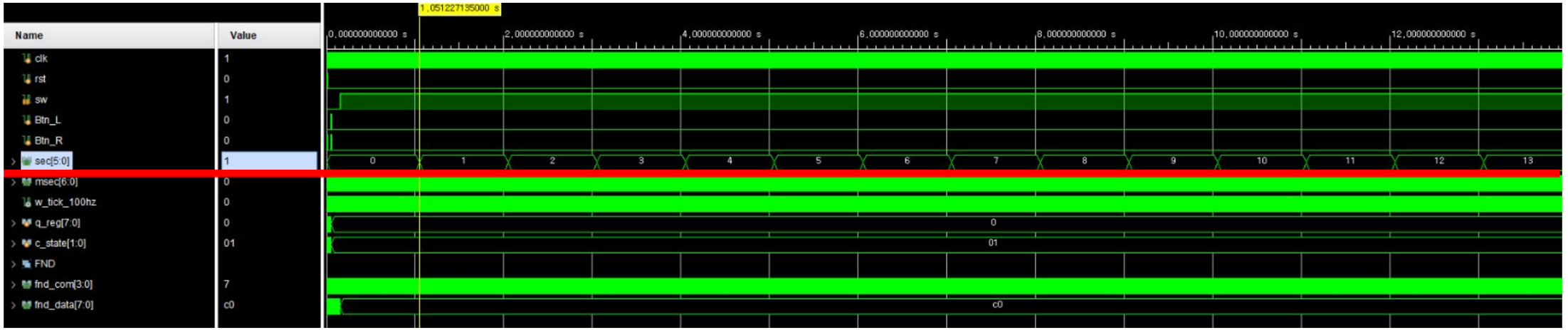

w_tick_100hz 가 1이 될 때마다 count하고 있다.

# 3. 시뮬레이션 (2)



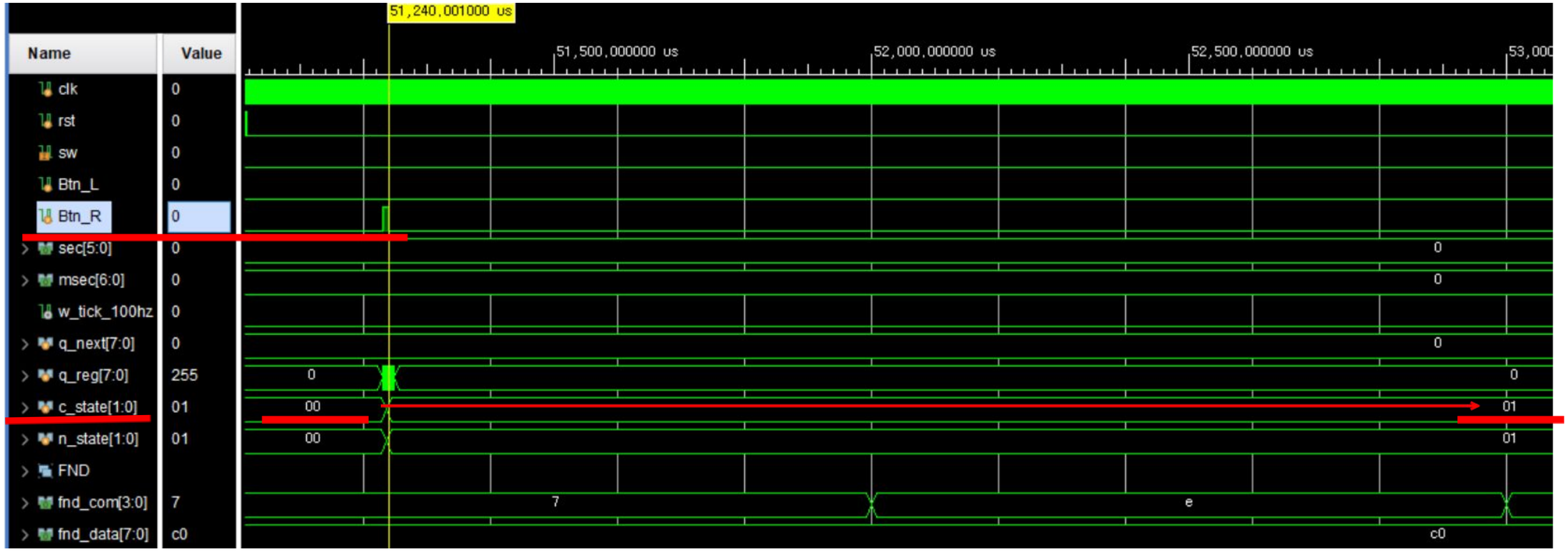

msec 가 99 가 되고 그 다음 클럭에서 sec 가 1 증가하며 msec 는 0으로 돌아간다.

# 3. 시뮬레이션 (3)



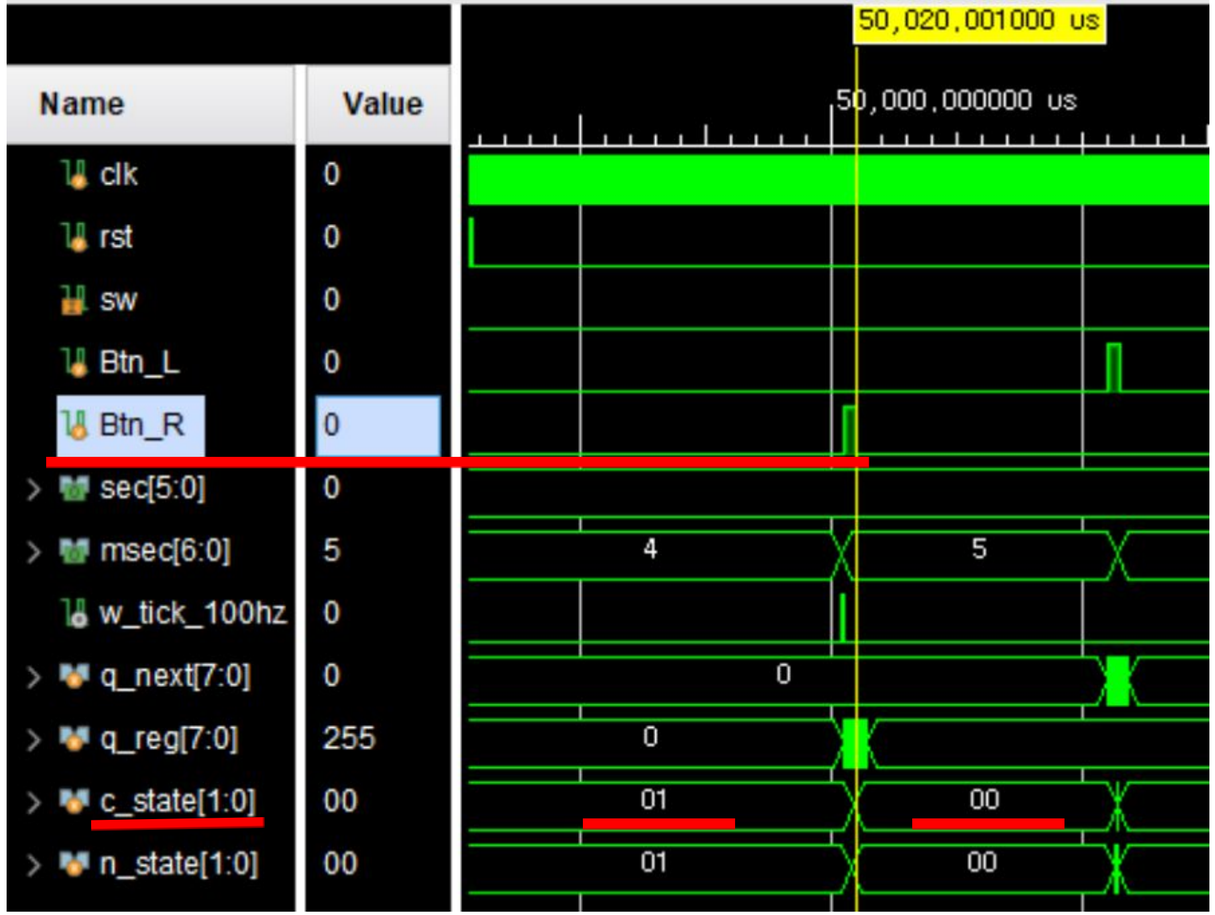

second도 count되며 증가하고 있다

# 3. 시뮬레이션 (4)



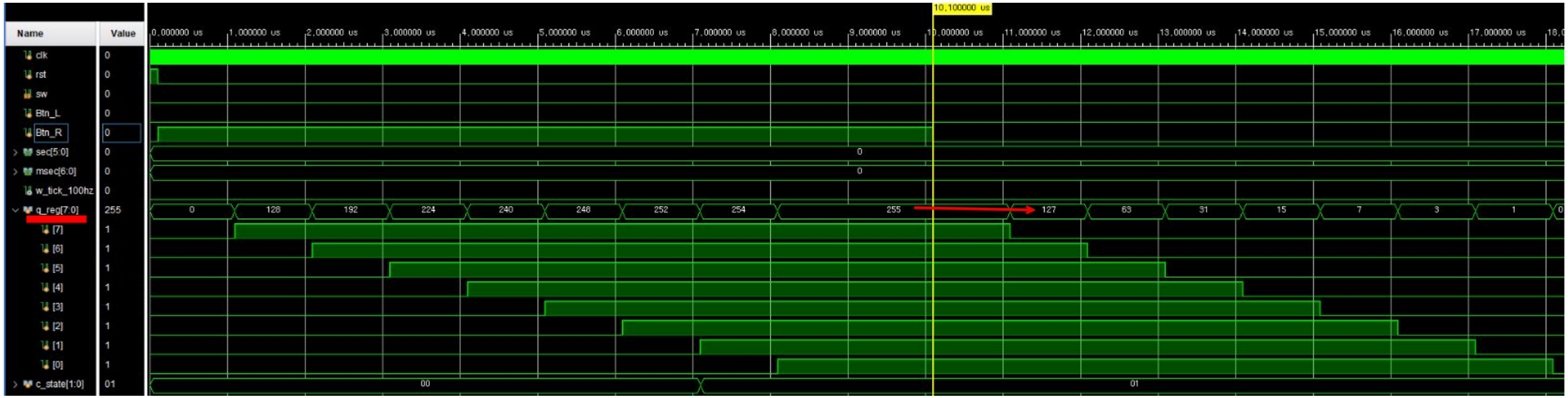

btn_R 입력으로 c_state 가 00(STOP)에서 01(RUN)으로 변한다.

# 3. 시뮬레이션 (5)



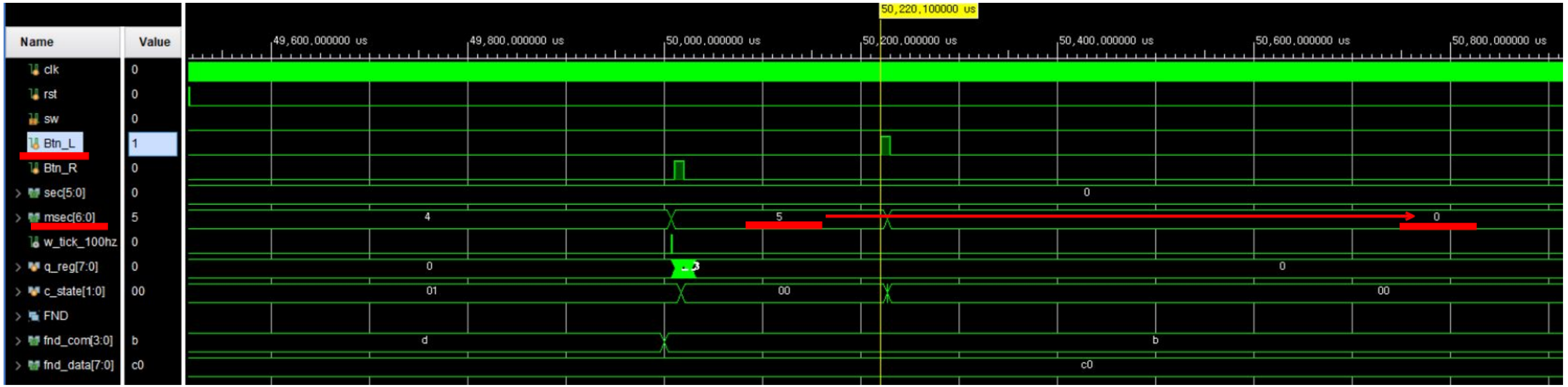

btn_R 입력으로 c_state 가 01(RUN)에서 00(STOP)으로 변한다.

# 3. 시뮬레이션 (6)



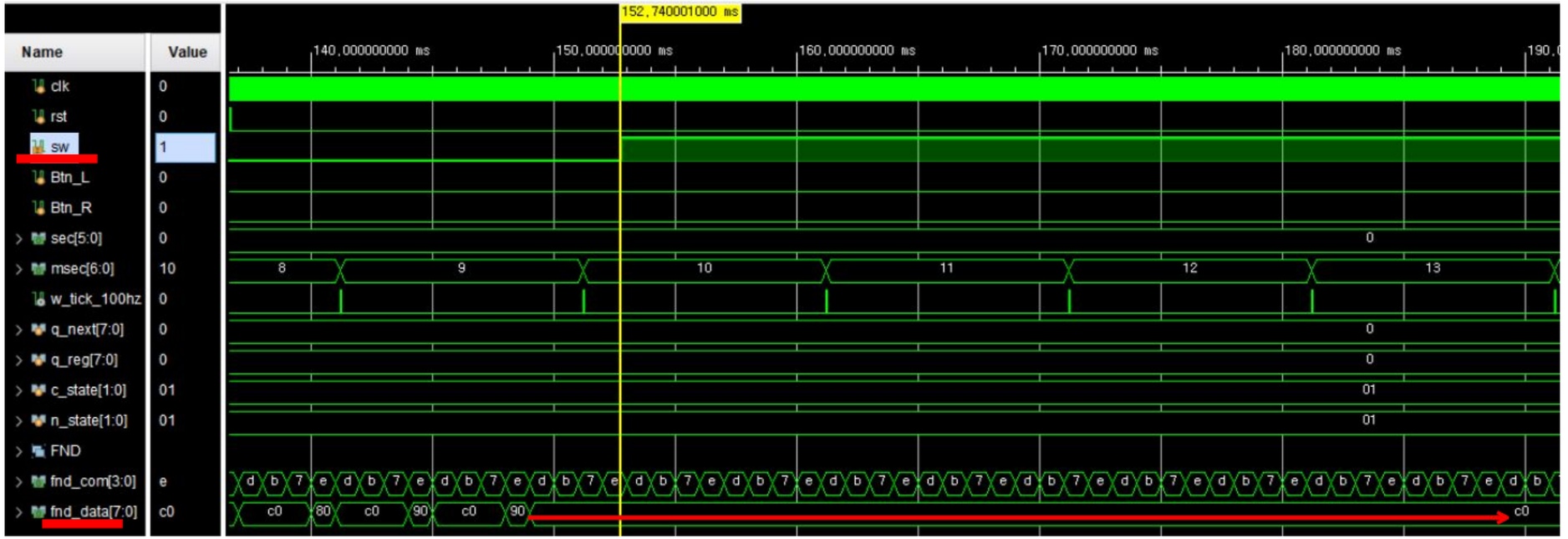

우측 버튼 입력(1)이 더이상 없으면 q_reg 값이 감소한다
-> bit 0이 들어오고 있다.

# 3. 시뮬레이션 (7)



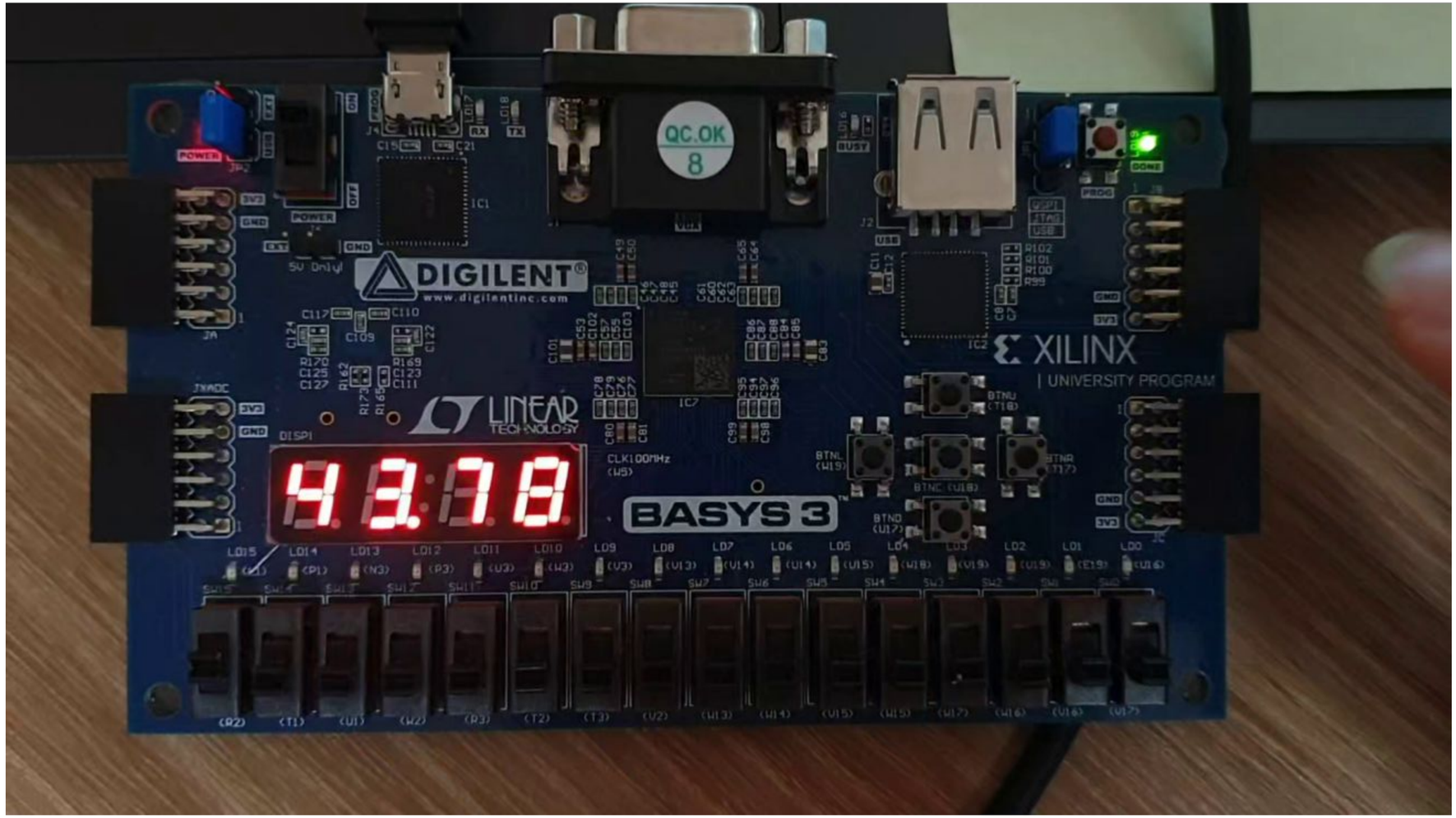

btn_L 입력으로 clear 되어 msec가 5에서 0으로 초기화된다.

# 3. 시뮬레이션 (8)




sw=1 이 되면 fnd_data 가 0이 된다.
sec.msec를 가리키던 화면에서 hour.min 화면으로 전환하기 때문

# 4. 동작 영상


QC.OK
8
DIGILENT
www.digilentinc.com
LINEAR TECHNOLOGY
XILINX
UNIVERSITY PROGRAM
BASYS 3
4 3 7 8

# 5. Trouble Shooting

1. 7-segment diplay 가 깜빡이는 문제

2. watch 모드에서 버튼 동작이 진행되지 않는다.

# Thank You!

경청해주셔서 감사합니다.