Basys3 Fusion Final Project

HW Design with VerilogHDL

Group No. 5

-장　환
-최현우
-윤종민
-권혁진

Presentation

발표 순서

1

HC-SR04

2

DHT11

3

Merge

목차

1
개요

2
상세설계

3
검증

4
결과

5
문제 해결

6
마무리

SR04

1. 개요
2. 상세설계
3. 검증
4. 결과
5. 문제해결

1.
개요

1. SPEC

개요

1.spec

-Board

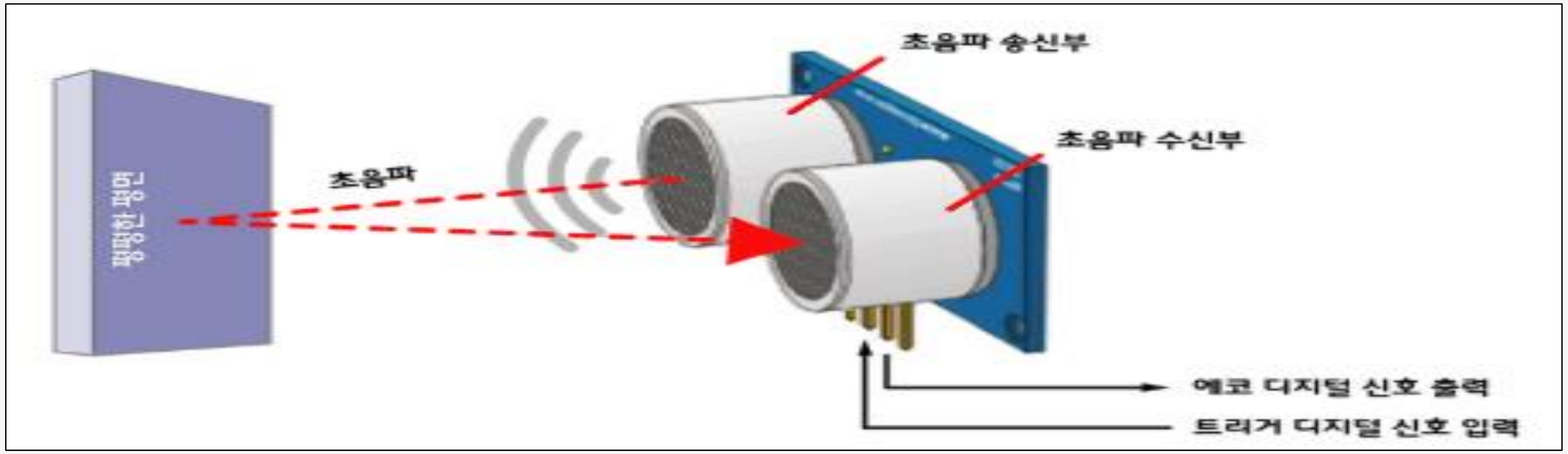
-HC-SR04

HC-SR04
-Operating Voltage : 5v(3.3v)
-Operating Current : 15mA
-Operating Frequency : 40KHz
-Measurement Distance : 2cm ~ 4m
-4 connect pins : VCC, GND, ECHO, TRIG

개요

1. SR-04 Project Goal

Ultrasonic sensor control



Summary

- 초음파 송신부(Trig), 수신부(Echo) 제어회로로 구성 되어 초음파를 제어해 거리 측정
- FND 거리 값 측정
- UART SENDER을 이용한 결과 값 확인

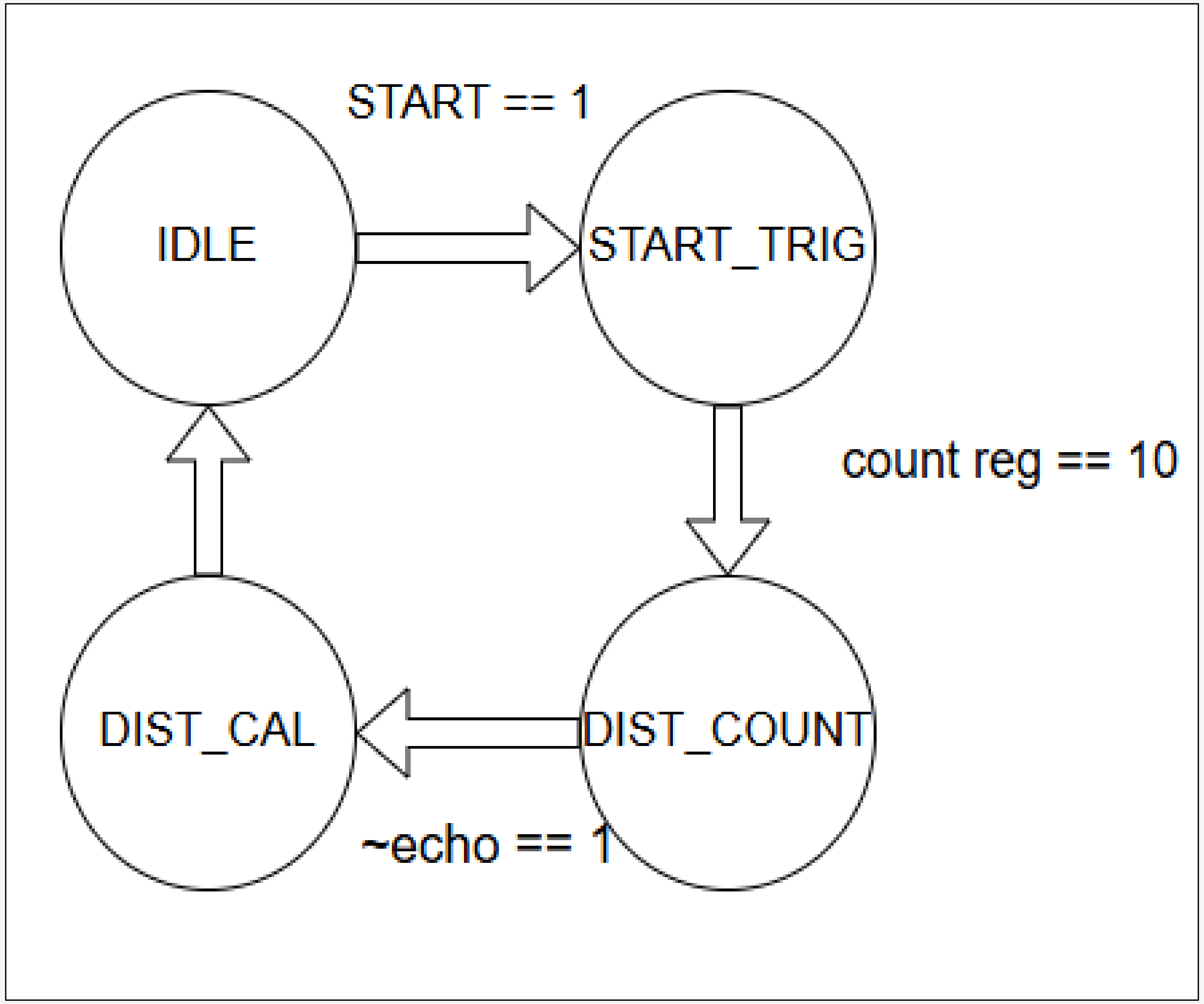
2.
상세설계

1. FSM&ASM
2. Block Diagram
3. Buttons & Switches

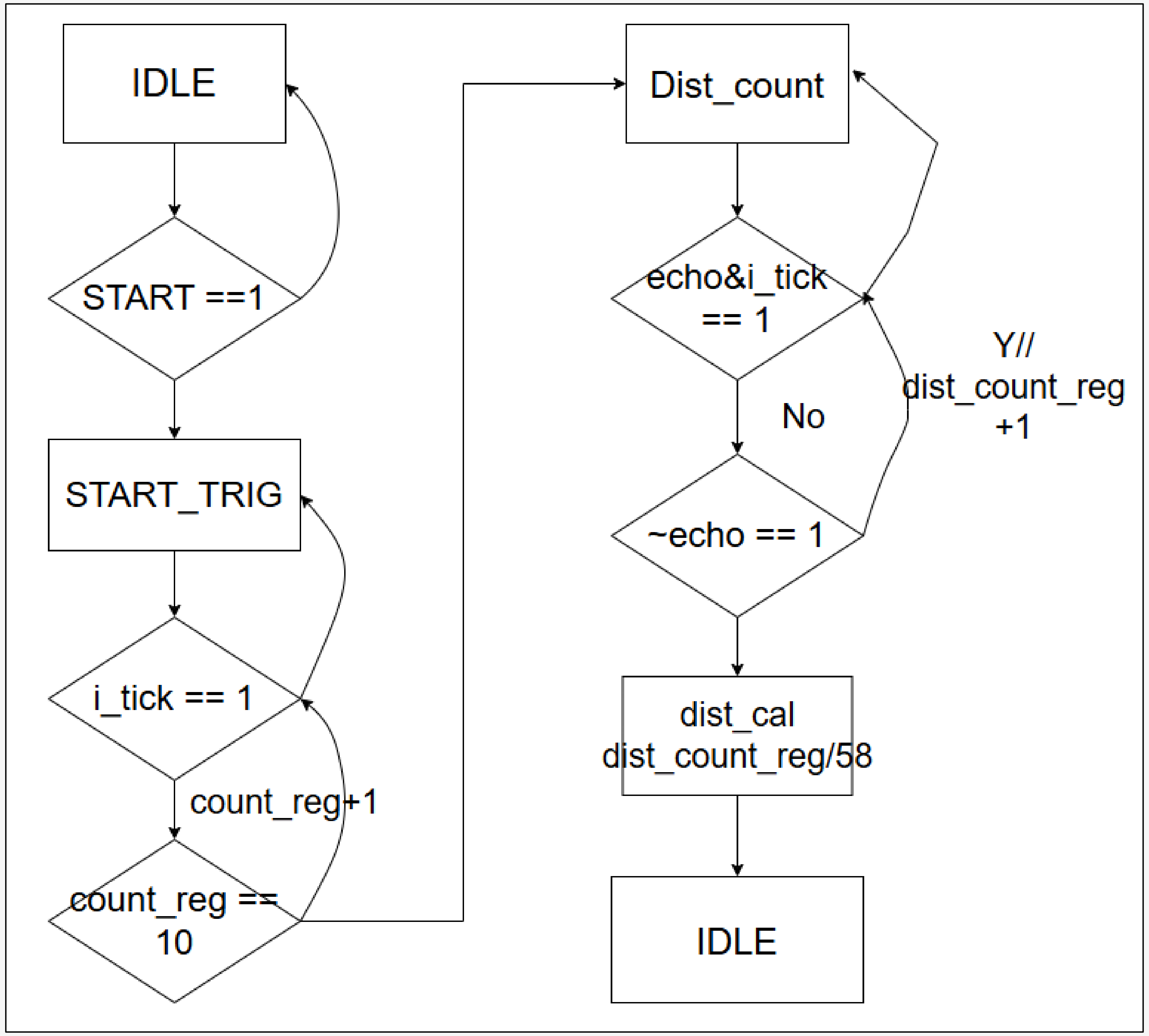
상세설계

1. FSM & ASM

FSM



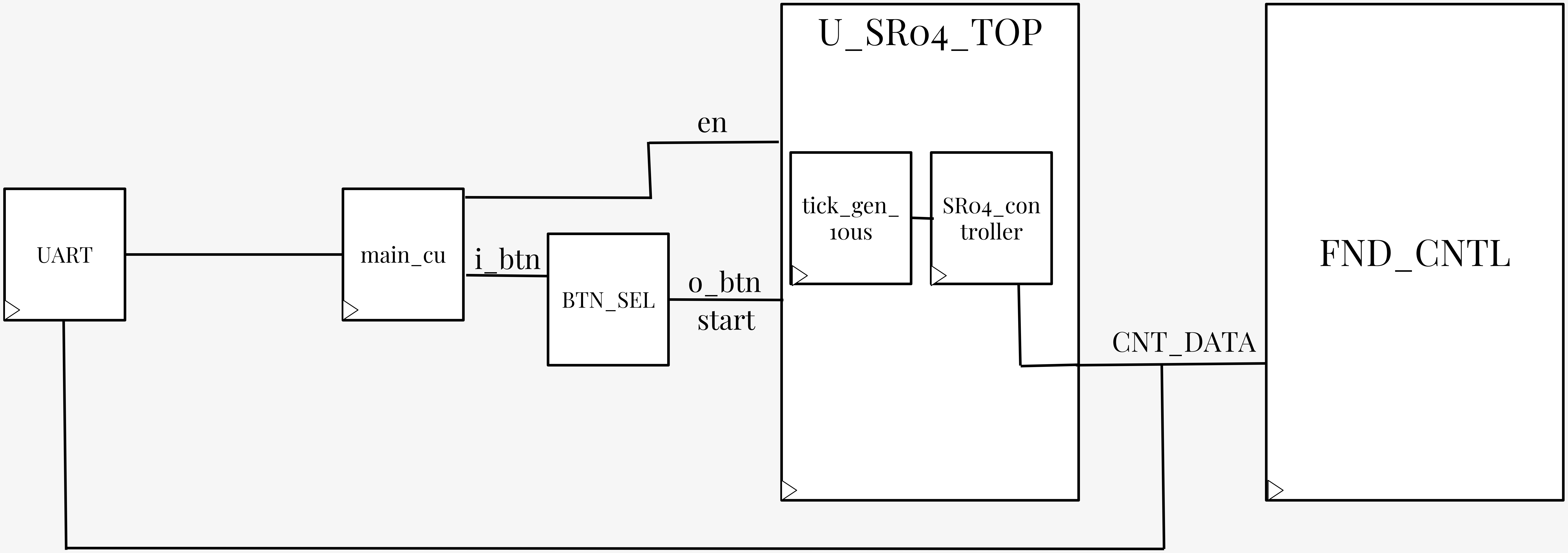
ASM



상세설계

2. Full HW Architecture

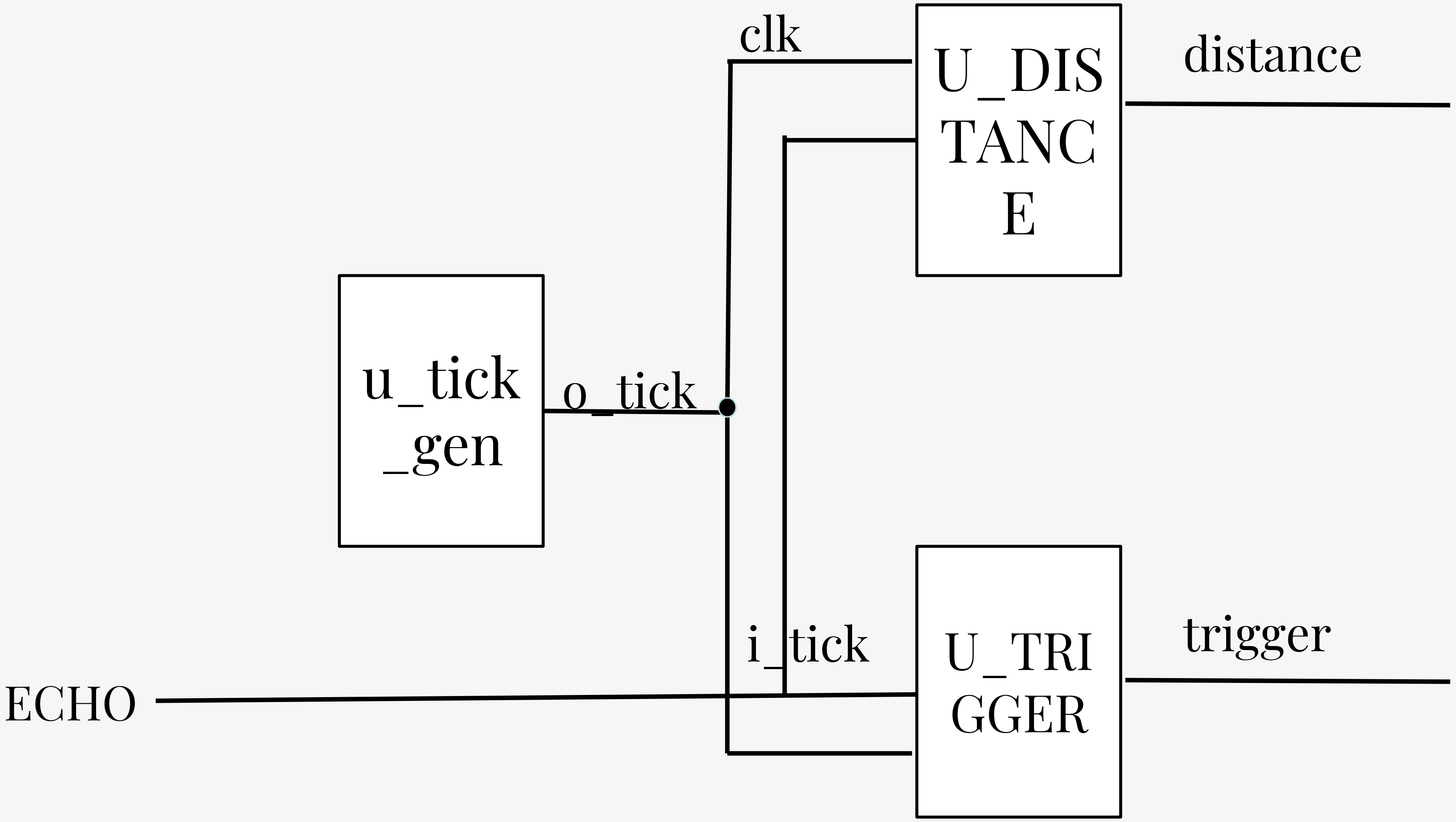
- SR04 Block Diagram



상세설계

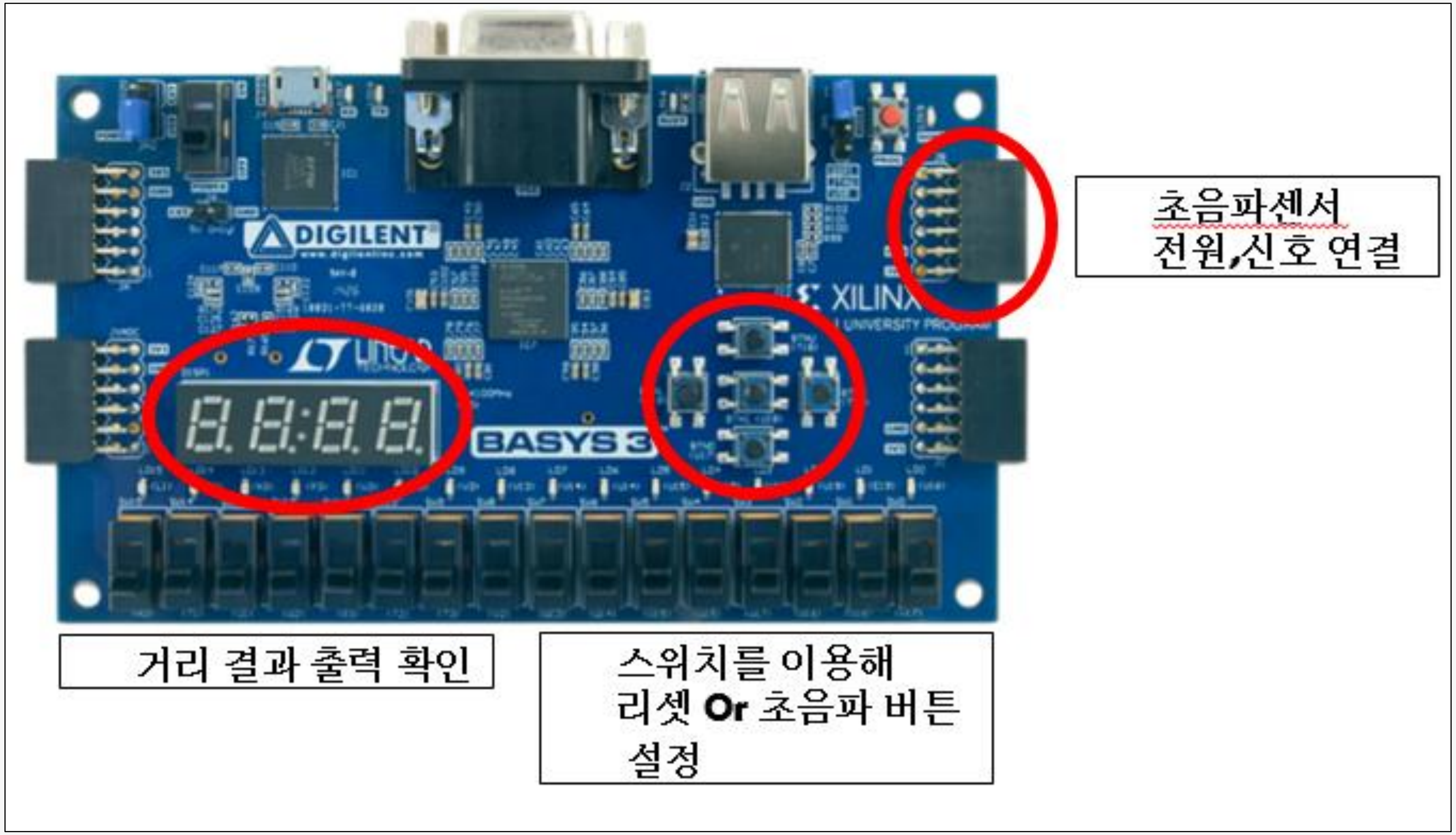
2. Block Diagram

- Controller Block Diagram



상세설계

3. Buttons & Switches



초음파센서
전원,신호 연결

거리 결과 출력 확인

스위치를 이용해
리셋 Or 초음파 버튼
설정

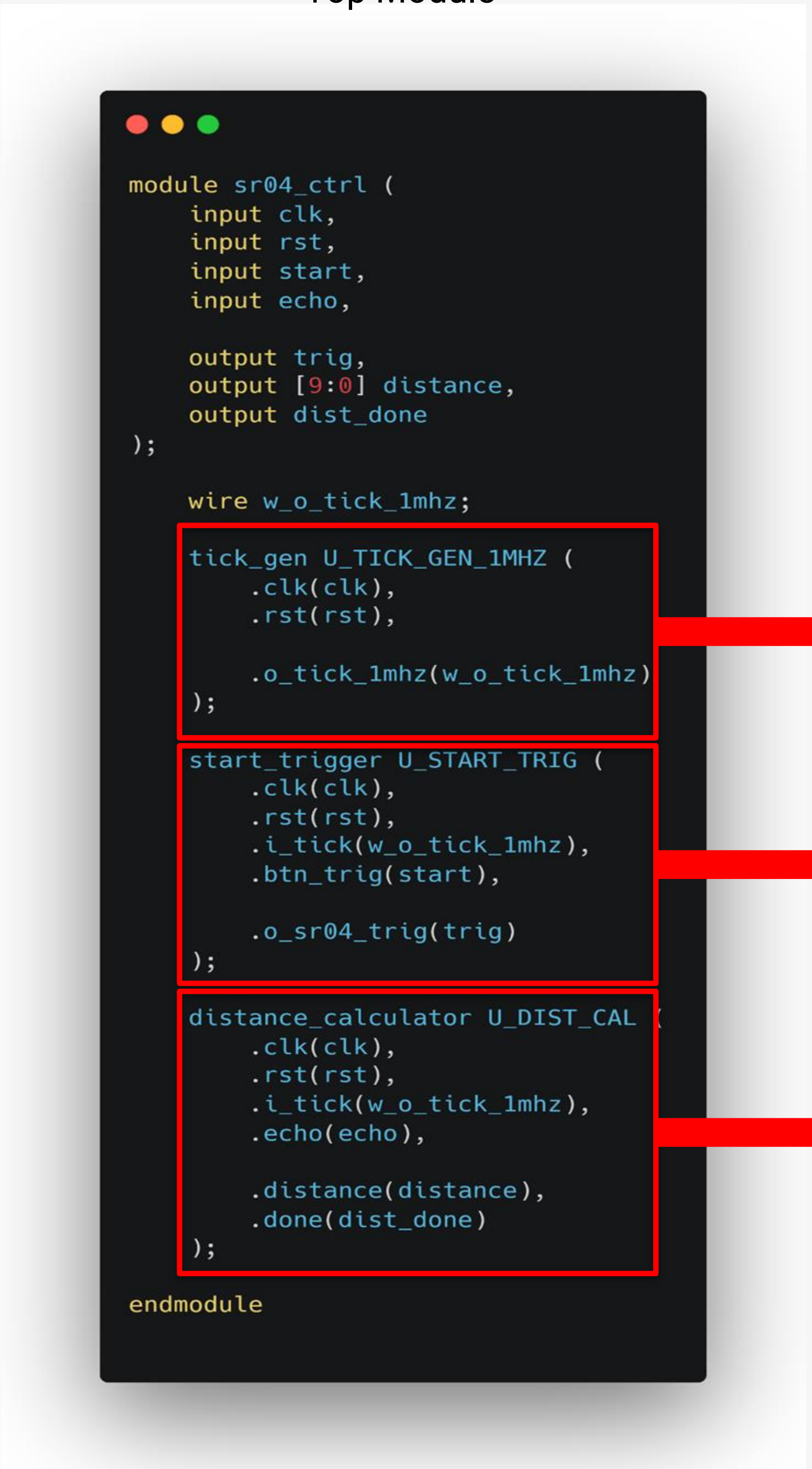
3.
검증

1. Code
2. Simulation

검증

1. Code

Top Module



```
module sr04_ctrl (
    input clk,
    input rst,
    input start,
    input echo,

    output trig,
    output [9:0] distance,
    output dist_done
);

    wire w_o_tick_1mhz;

    tick_gen U_TICK_GEN_1MHZ (
        .clk(clk),
        .rst(rst),

        .o_tick_1mhz(w_o_tick_1mhz)
    );

    start_trigger U_START_TRIG (
        .clk(clk),
        .rst(rst),
        .i_tick(w_o_tick_1mhz),
        .btn_trig(start),

        .o_sr04_trig(trig)
    );

    distance_calculator U_DIST_CAL (
        .clk(clk),
        .rst(rst),
        .i_tick(w_o_tick_1mhz),
        .echo(echo),

        .distance(distance),
        .done(dist_done)
    );

endmodule
```

1Mhz Tick 생성기

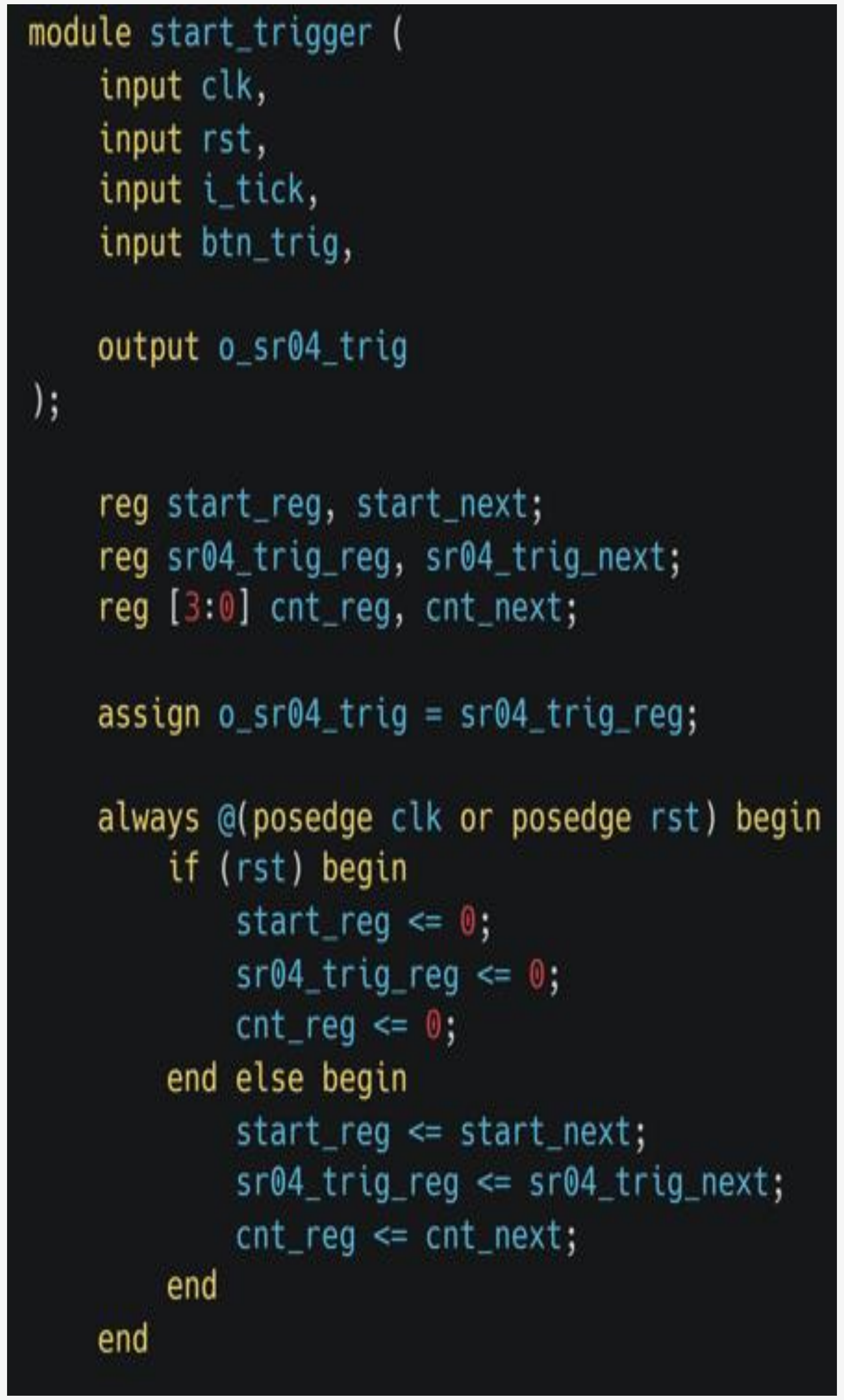
Trig Pulse 생성기

Distance(거리) 계산기

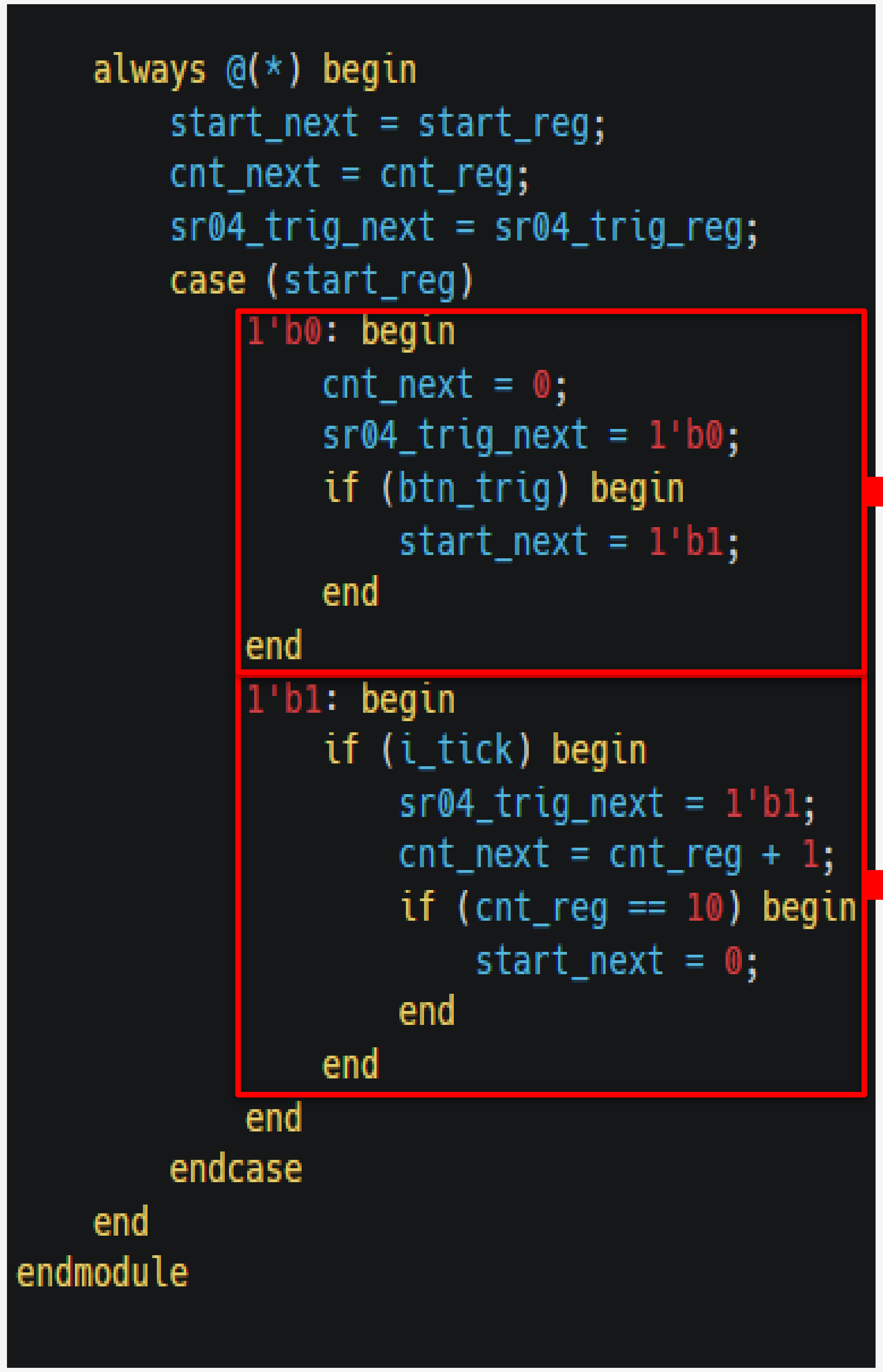
검증

1. Code

Start Tirg Module



```
module start_trigger (
    input clk,
    input rst,
    input i_tick,
    input btn_trig,

    output o_sr04_trig
);


    reg start_reg, start_next;
    reg sr04_trig_reg, sr04_trig_next;
    reg [3:0] cnt_reg, cnt_next;

    assign o_sr04_trig = sr04_trig_reg;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            start_reg <= 0;
            sr04_trig_reg <= 0;
            cnt_reg <= 0;
        end else begin
            start_reg <= start_next;
            sr04_trig_reg <= sr04_trig_next;
            cnt_reg <= cnt_next;
        end
    end
```



```
    always @(*) begin
        start_next = start_reg;
        cnt_next = cnt_reg;
        sr04_trig_next = sr04_trig_reg;
        case (start_reg)
            1'b0: begin
                cnt_next = 0;
                sr04_trig_next = 1'b0;
                if (btn_trig) begin
                    start_next = 1'b1;
                end
            end
            1'b1: begin
                if (i_tick) begin
                    sr04_trig_next = 1'b1;
                    cnt_next = cnt_reg + 1;
                    if (cnt_reg == 10) begin
                        start_next = 0;
                    end
                end
            end
        endcase
    end
endmodule
```

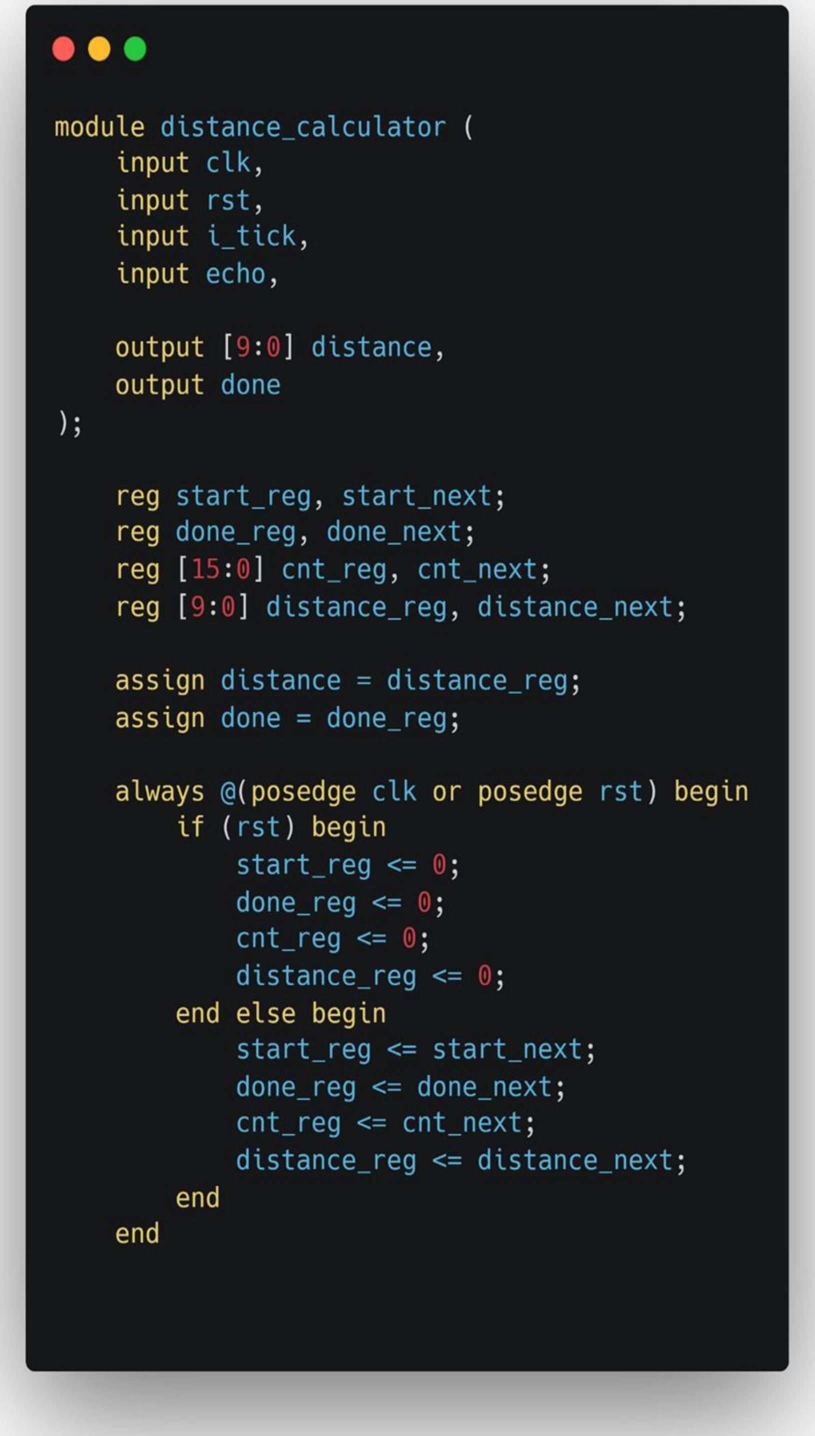
IDLE 상태

TRIG 발생 상태

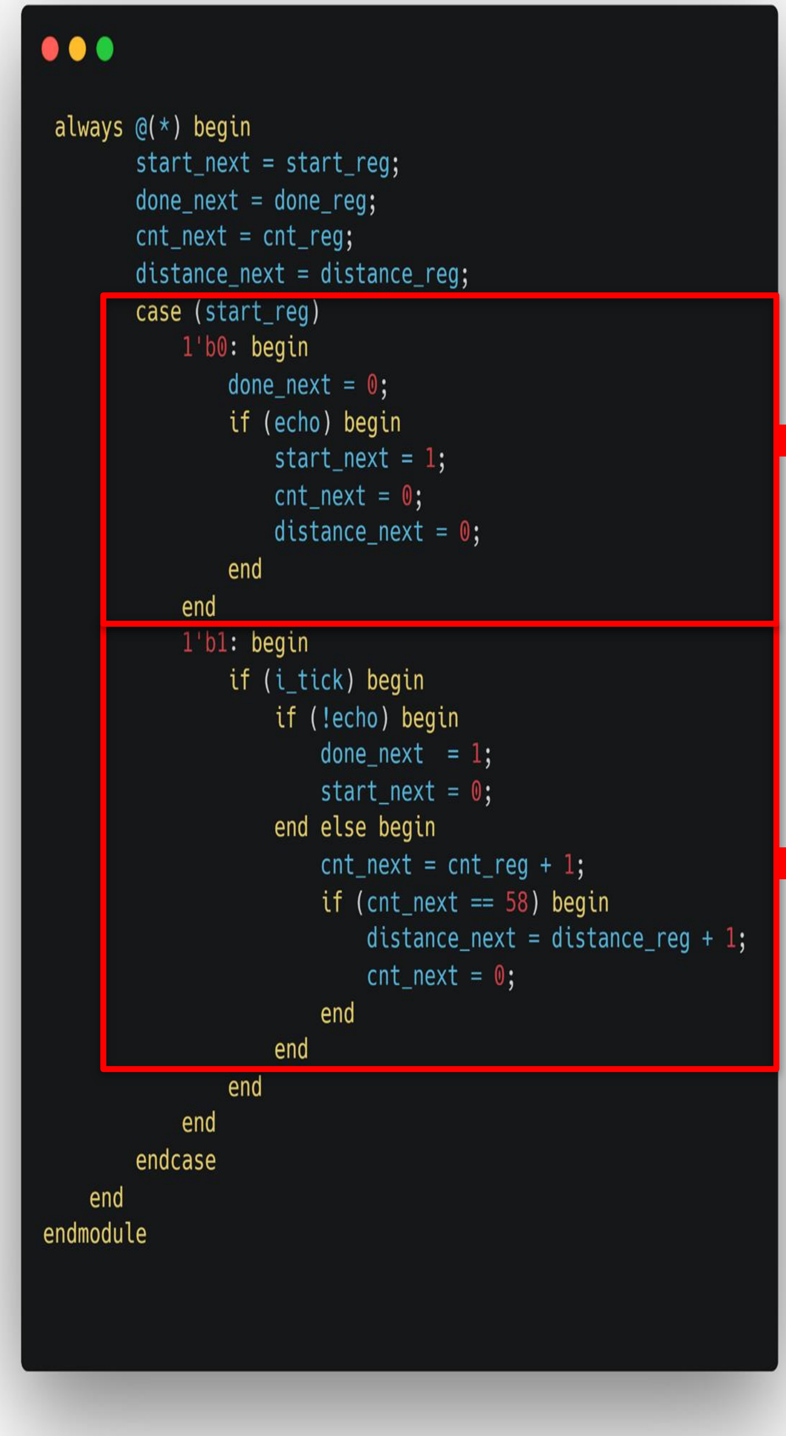
검증

1. Code

Distance Module



```
module distance_calculator (
    input clk,
    input rst,
    input i_tick,
    input echo,

    output [9:0] distance,
    output done
);

    reg start_reg, start_next;
    reg done_reg, done_next;
    reg [15:0] cnt_reg, cnt_next;
    reg [9:0] distance_reg, distance_next;

    assign distance = distance_reg;
    assign done = done_reg;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            start_reg <= 0;
            done_reg <= 0;
            cnt_reg <= 0;
            distance_reg <= 0;
        end else begin
            start_reg <= start_next;
            done_reg <= done_next;
            cnt_reg <= cnt_next;
            distance_reg <= distance_next;
        end
    end
```



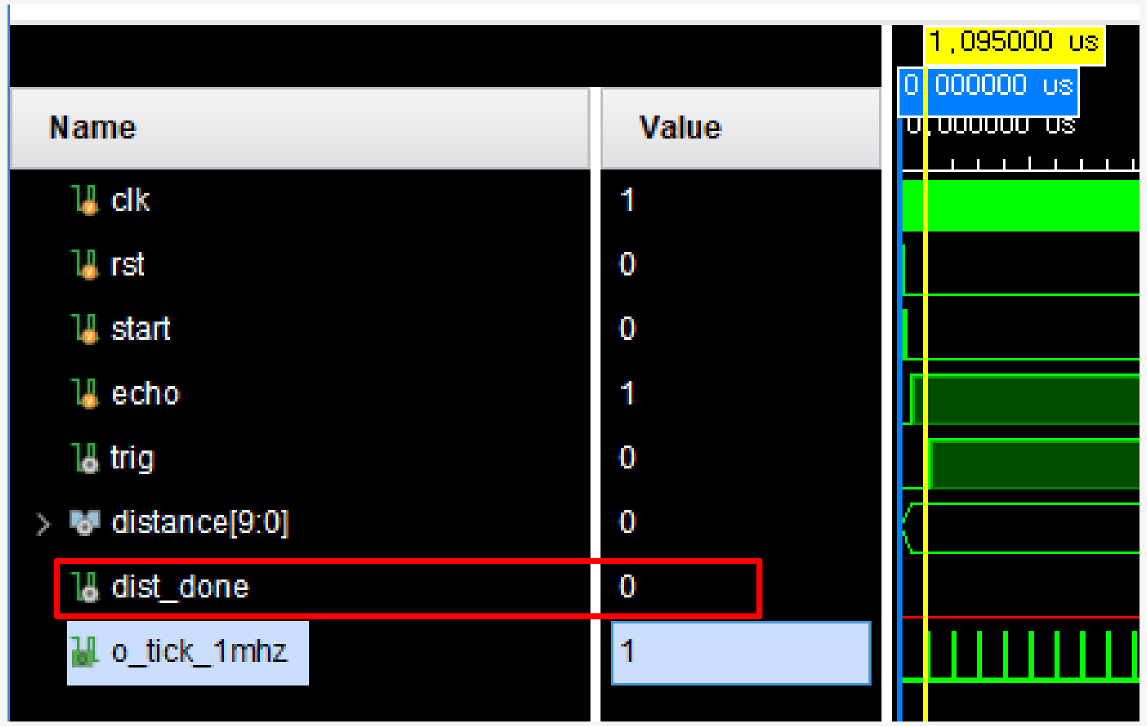
```
    always @(*) begin
        start_next = start_reg;
        done_next = done_reg;
        cnt_next = cnt_reg;
        distance_next = distance_reg;
        case (start_reg)
            1'b0: begin
                done_next = 0;
                if (echo) begin
                    start_next = 1;
                    cnt_next = 0;
                    distance_next = 0;
                end
            end
            1'b1: begin
                if (i_tick) begin
                    if (!echo) begin
                        done_next  = 1;
                        start_next = 0;
                    end else begin
                        cnt_next = cnt_reg + 1;
                        if (cnt_next == 58) begin
                            distance_next = distance_reg + 1;
                            cnt_next = 0;
                        end
                    end
                end
            end
        endcase
    end
endmodule
```

IDLE 상태

MUSURING 상태

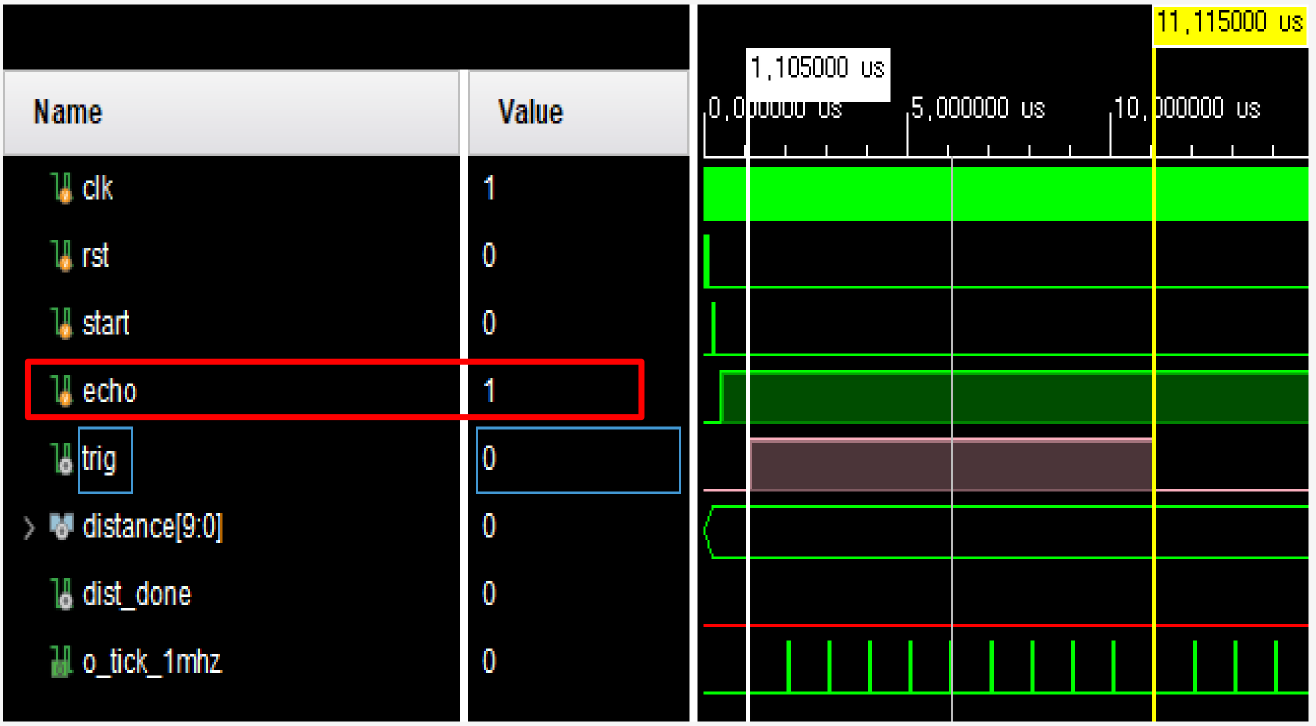
검증

2. simulation



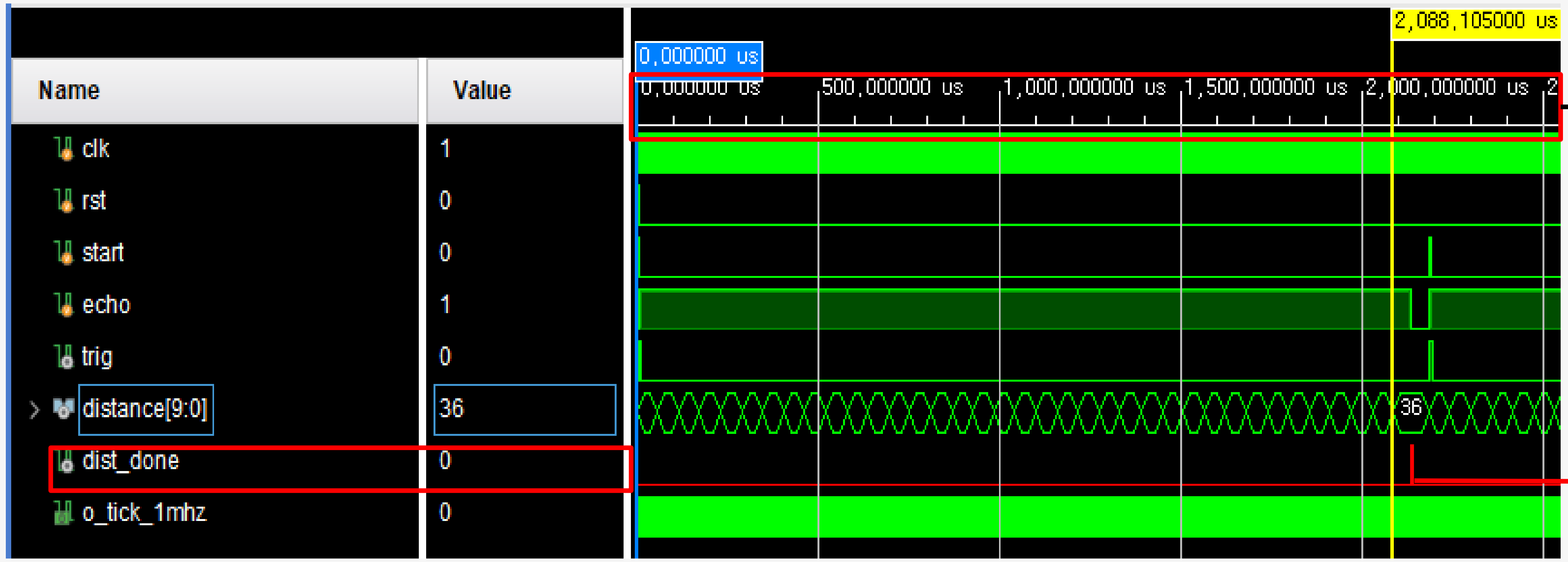
Name	Value
clk	1
rst	0
start	0
echo	1
trig	0
distance[9:0]	0
dist_done	0
o_tick_1mhz	1

tick 간격은 1us simulation에서도 됨



Name	Value
clk	1
rst	0
start	0
echo	1
trig	0
distance[9:0]	0
dist_done	0
o_tick_1mhz	0

trig또한 10us동안 지속



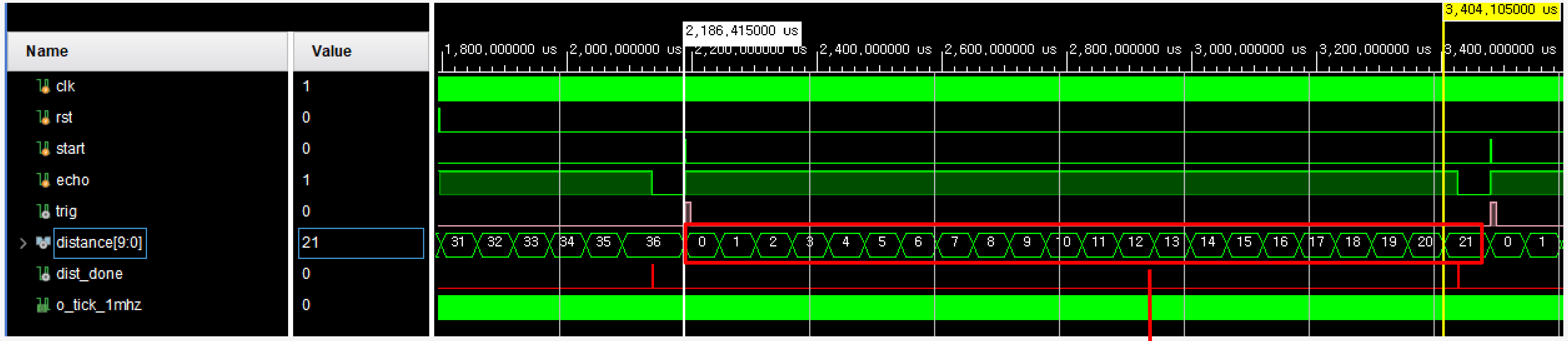
Name	Value
clk	1
rst	0
start	0
echo	1
trig	0
distance[9:0]	36
dist_done	0
o_tick_1mhz	0

2088/58 == 36 distance또한 잘 측정되고 있다

측정 종료 후 dist_done신호도 이상 무

검증

2. simulation



Name	Value
clk	1
rst	0
start	0
echo	1
trig	0
distance[9:0]	21
dist_done	0
o_tick_1mhz	0

랜덤값으로 시뮬레이션 돌린 결과

3404-2186 = 1217

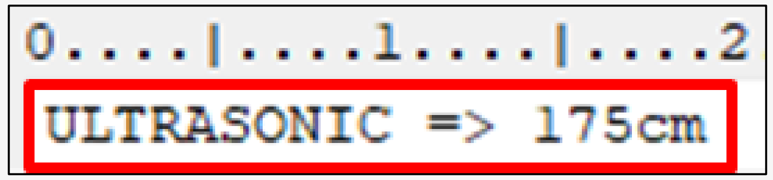
1218*58 = 21

4. 결과

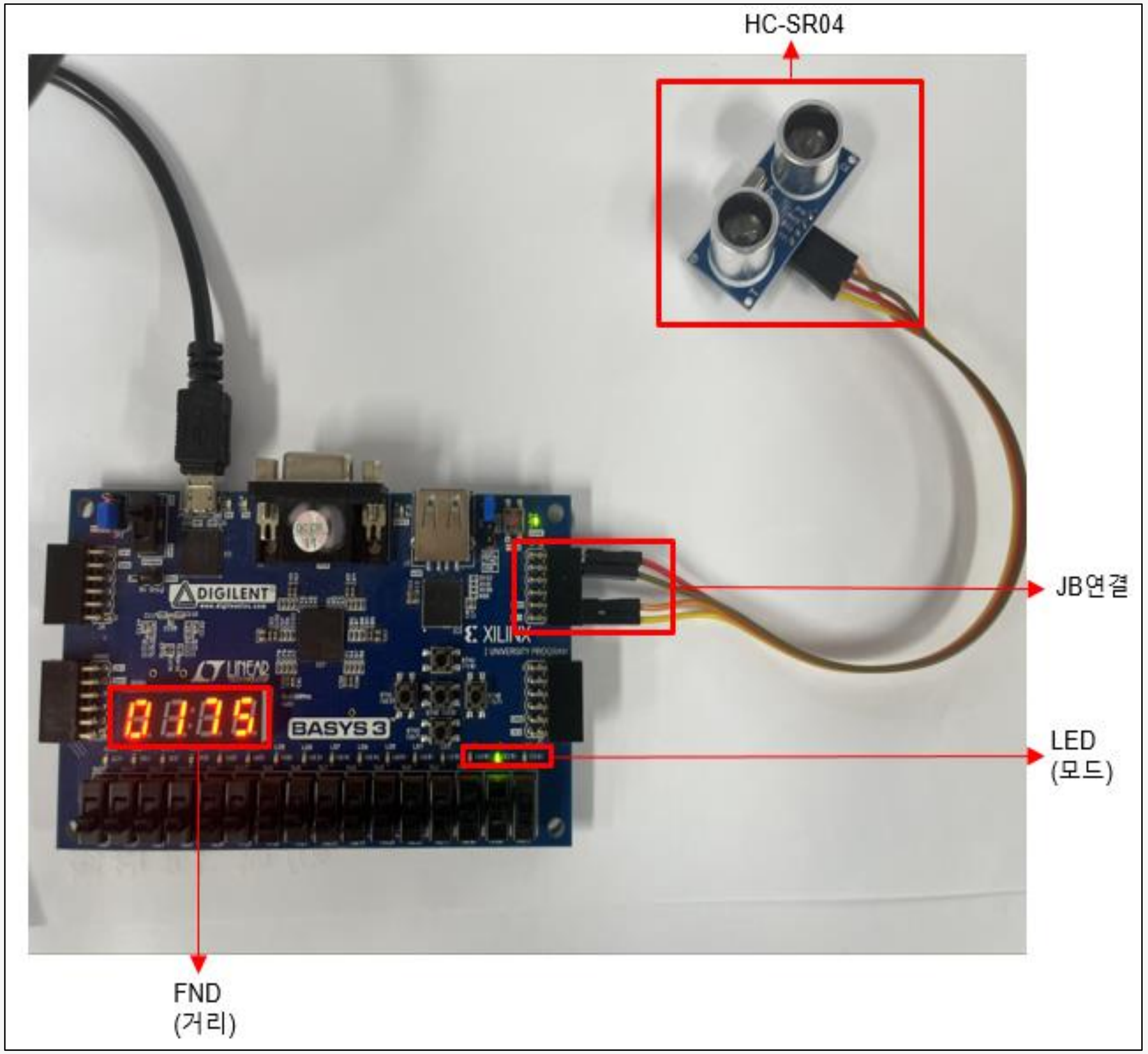
1. Summary

결과

1. SR-04 Features Summary



```
0....|....1....|....2
ULTRASONIC => 175cm
```



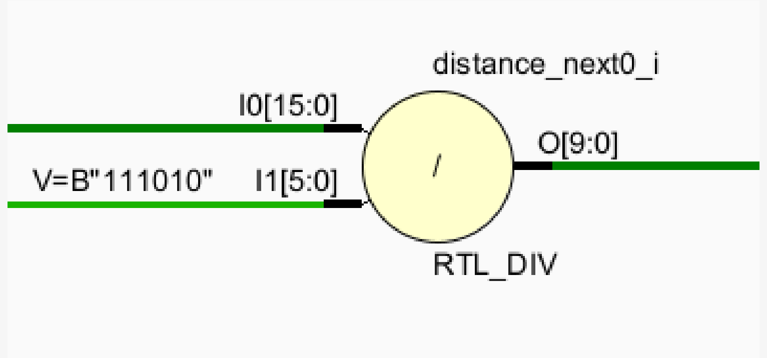
5.
문제 해결

1. Problem & Solution

문제해결

58 Count Divider Slack

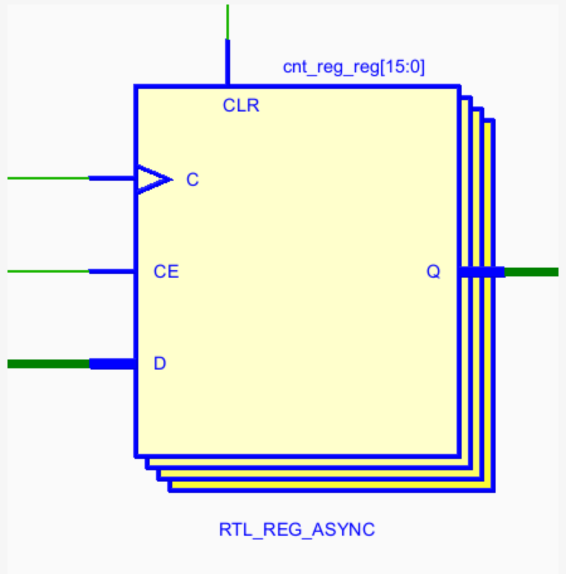
Problem(Divider)



Long Critical Path

→ Negative Slack

Solution(Counter)



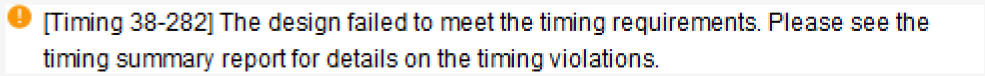
Decrease Critical Path

→ Solve Slack

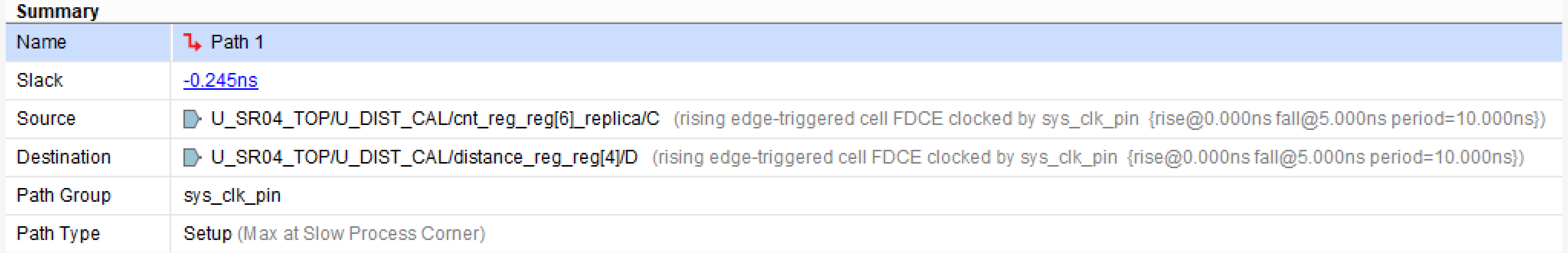
→ 58 Tick 나눗셈기를 Counter로 해결

검증

1. code

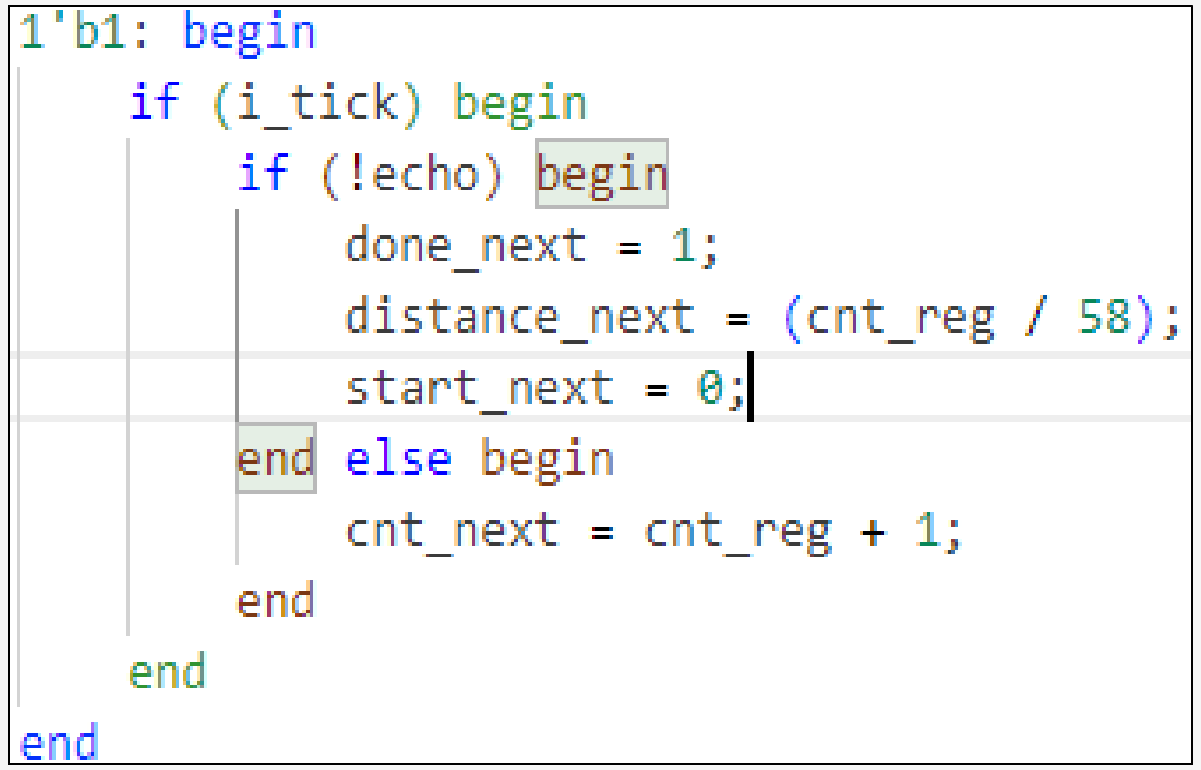


[Timing 38-282] The design failed to meet the timing requirements. Please see the timing summary report for details on the timing violations.



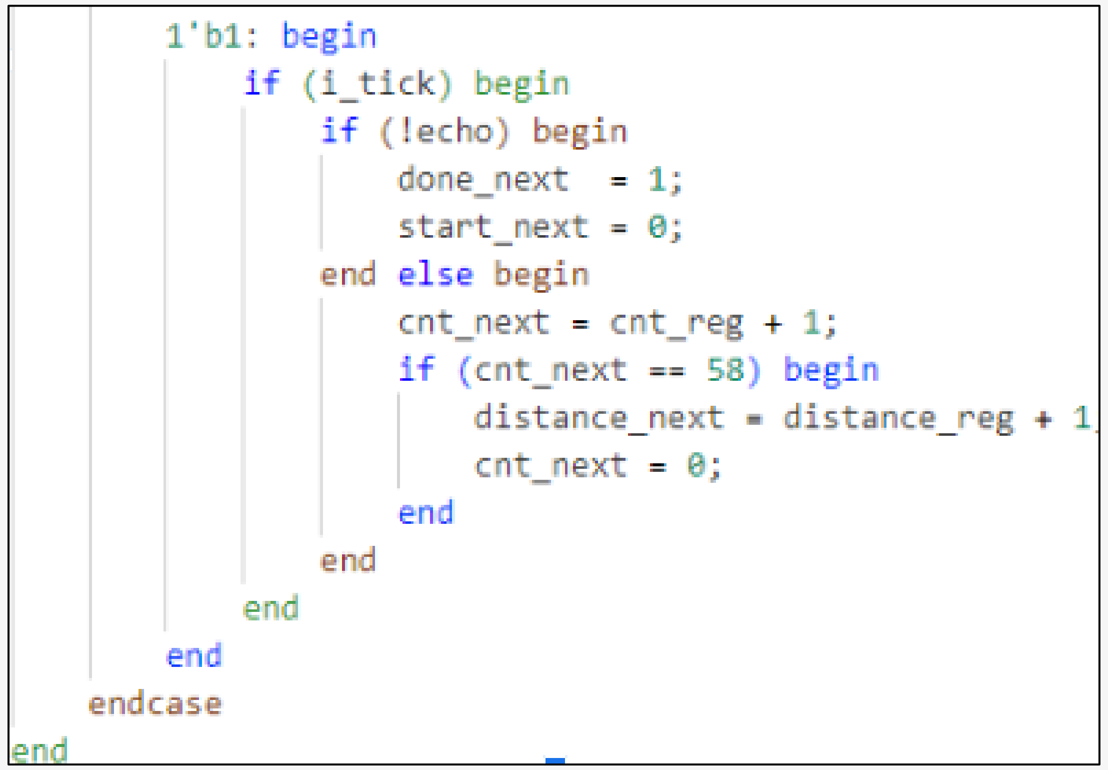
Summary

Name	Path 1
Slack	-0.245ns
Source	U_SR04_TOP/U_DIST_CAL/cnt_reg_reg[6]_replica/C (rising edge-triggered cell FDCE clocked by sys_clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})
Destination	U_SR04_TOP/U_DIST_CAL/distance_reg_reg[4]/D (rising edge-triggered cell FDCE clocked by sys_clk_pin {rise@0.000ns fall@5.000ns period=10.000ns})
Path Group	sys_clk_pin
Path Type	Setup (Max at Slow Process Corner)



```
1'b1: begin
    if (i_tick) begin
        if (!echo) begin
            done_next = 1;
            distance_next = (cnt_reg / 58);
            start_next = 0;
        end else begin
            cnt_next = cnt_reg + 1;
        end
    end
end
```

<Before improvement>



```
        1'b1: begin
            if (i_tick) begin
                if (!echo) begin
                    done_next  = 1;
                    start_next = 0;
                end else begin
                    cnt_next = cnt_reg + 1;
                    if (cnt_next == 58) begin
                        distance_next = distance_reg + 1
                        cnt_next = 0;
                    end
                end
            end
        end
    endcase
end
```

<After improvement>

→ 58 Tick 나눗셈기를 Counter로 해결한 코드

DHT11

1. 개요
2. 상세설계
3. 검증
4. 결과
5. 문제해결
6. 고찰

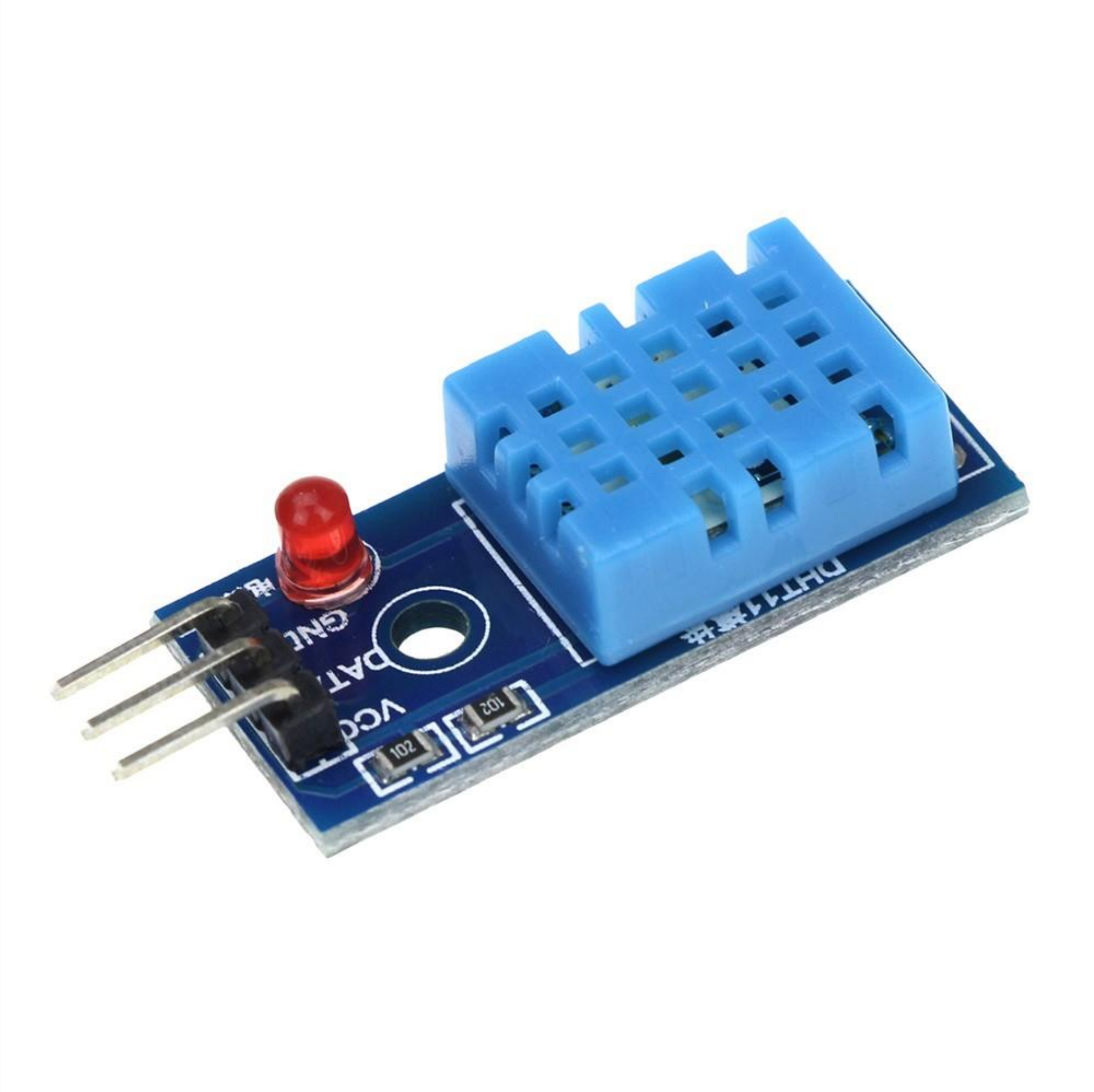
개요

1.spec

-Board

-DHT11

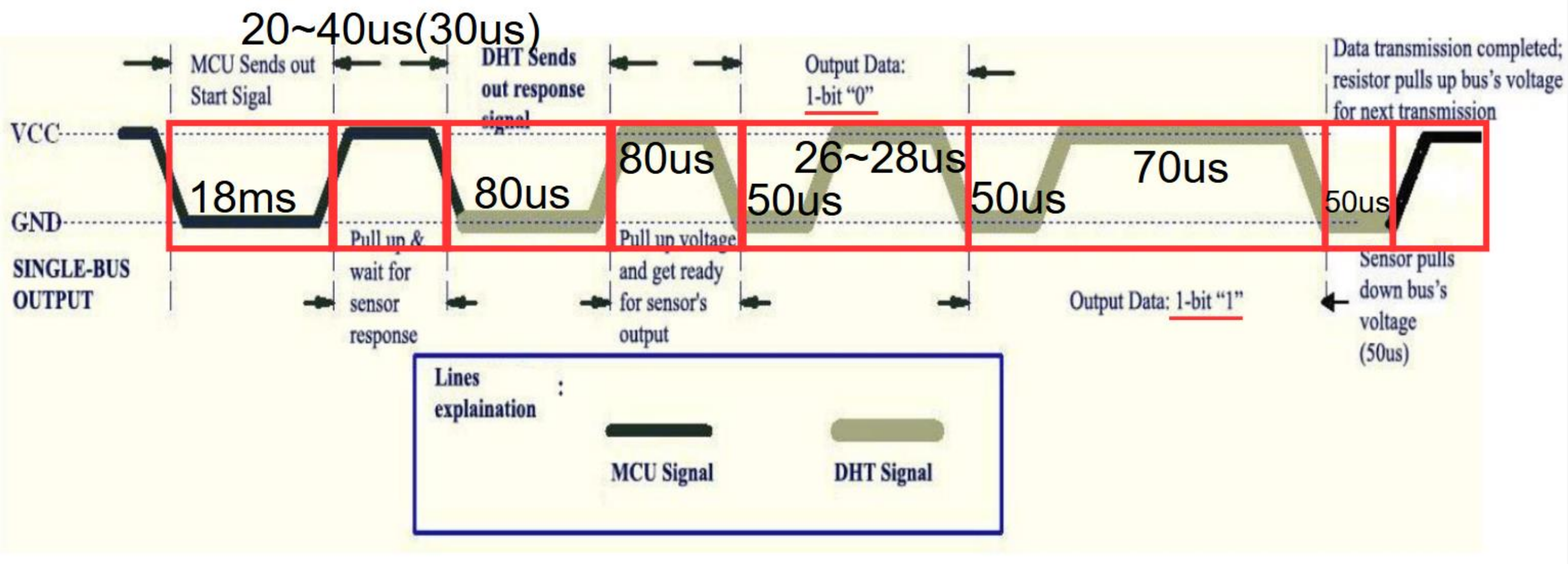
DHT11
-3 pins
-Temperature & Humidity sensor
-1 Led



개요

1. DHT11 Project Goal

DHT11 sensor control



20~40us(30us)
MCU Sends out Start Sigal
DHT Sends out response signal
Output Data: 1-bit "0"
Data transmission completed; resistor pulls up bus's voltage for next transmission
VCC
GND
SINGLE-BUS OUTPUT
18ms
80us
80us
50us
26~28us
50us
70us
50us
Pull up & wait for sensor response
Pull up voltage and get ready for sensor's output
Output Data: 1-bit "1"
Sensor pulls down bus's voltage (50us)
Lines explaination :
MCU Signal
DHT Signal

Summary

- 수분으로 인한 전극간 전류로 습도 측정
- 온도에 따라 변하는 저항값(서미스터)로 온도 측정
- Half duplex communication
- 결과 값은 FND 표기
- UART SENDER을 이용한 결과 값 확인

8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum

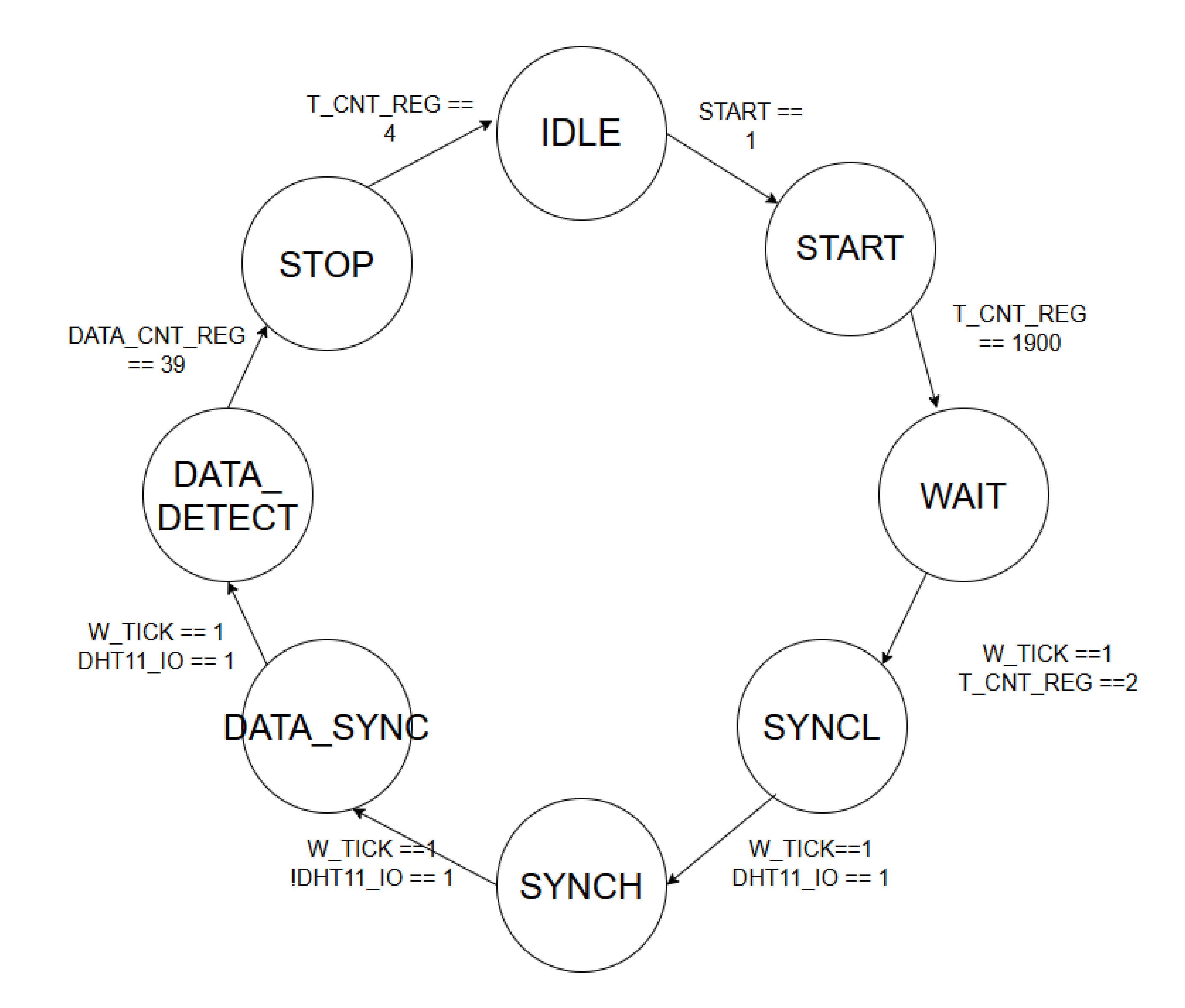
2.
상세설계

1. FSM&ASM
2. Block Diagram
3. Buttons & Switches

상세설계

1. FSM & ASM

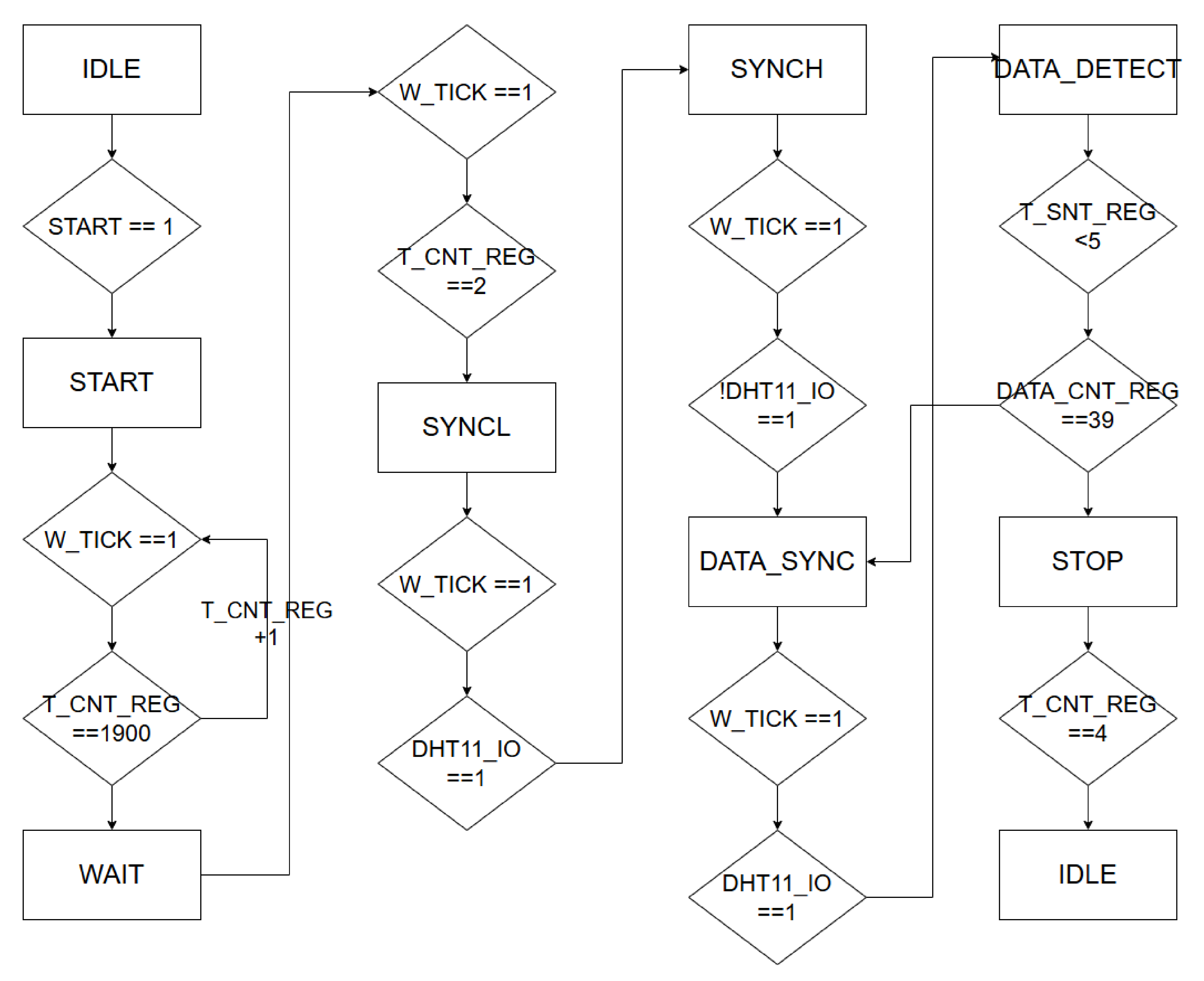
FSM



상세설계

1. FSM & ASM

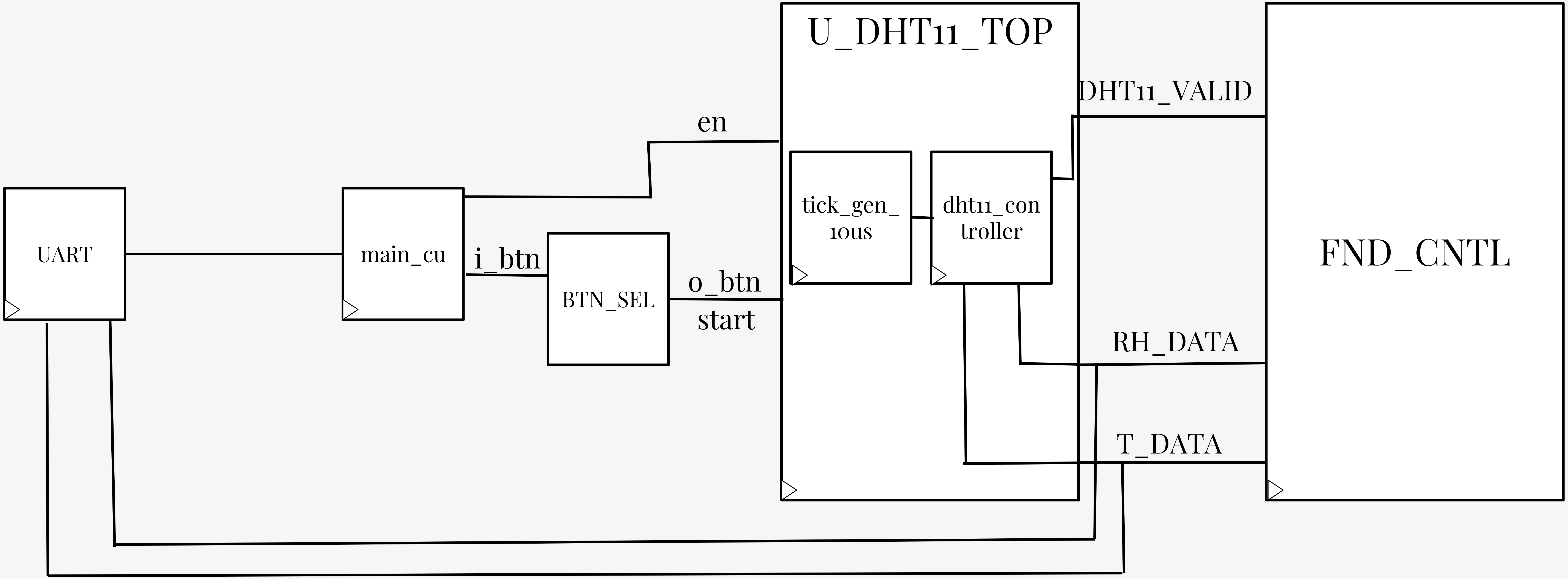
ASM



상세설계

2. Full HW Architecture

- DHT11 Block Diagram

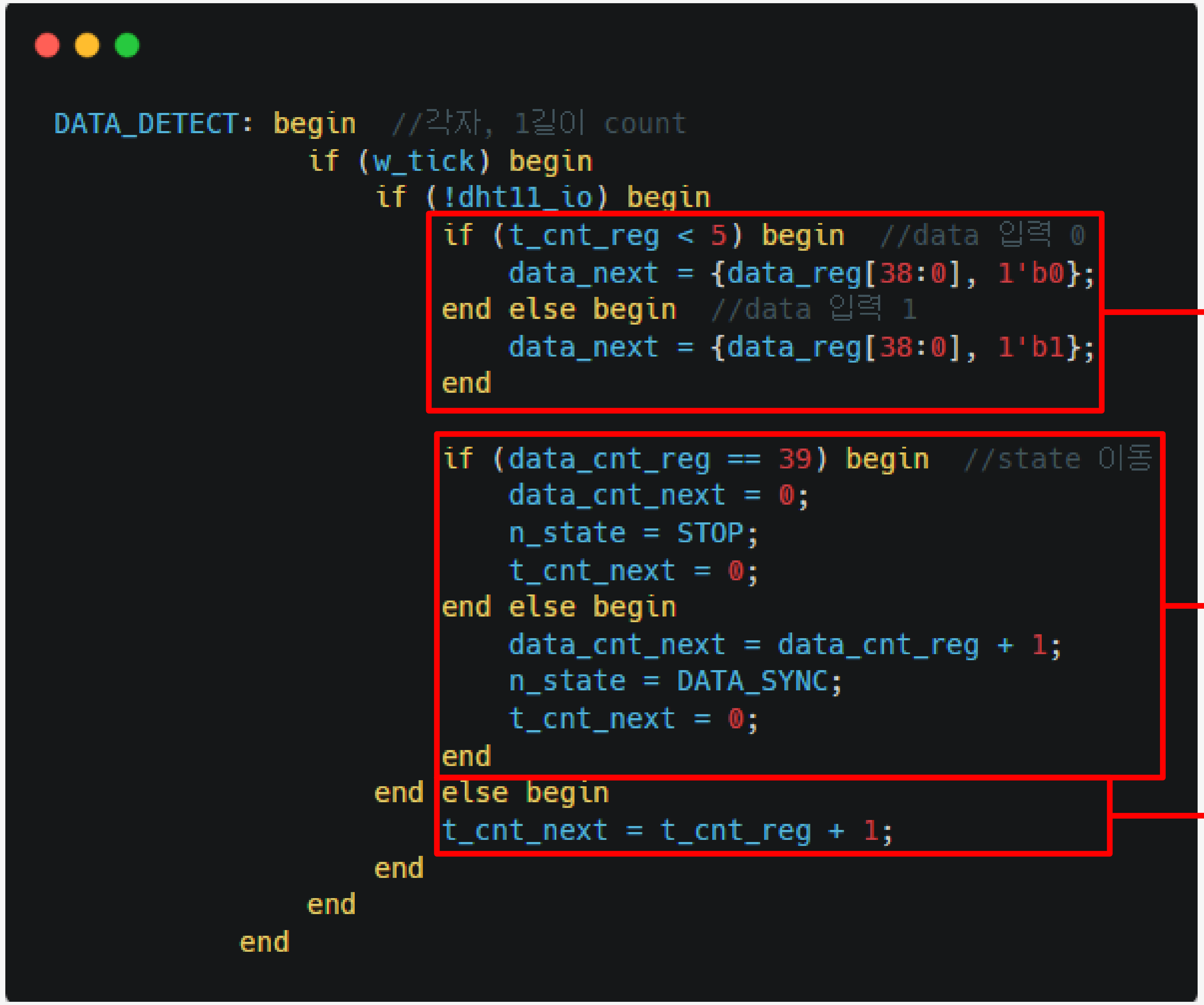


3.검증

1. Code
2. Simulation

검증

code



```
DATA_DETECT: begin  //각자, 1길이 count
               if (w_tick) begin
                   if (!dht11_io) begin
                       if (t_cnt_reg < 5) begin  //data 입력 0
                           data_next = {data_reg[38:0], 1'b0};
                       end else begin  //data 입력 1
                           data_next = {data_reg[38:0], 1'b1};
                       end

                       if (data_cnt_reg == 39) begin  //state 이동
                           data_cnt_next = 0;
                           n_state = STOP;
                           t_cnt_next = 0;
                       end else begin
                           data_cnt_next = data_cnt_reg + 1;
                           n_state = DATA_SYNC;
                           t_cnt_next = 0;
                       end
                   end else begin
                       t_cnt_next = t_cnt_reg + 1;
                   end
               end
           end
```

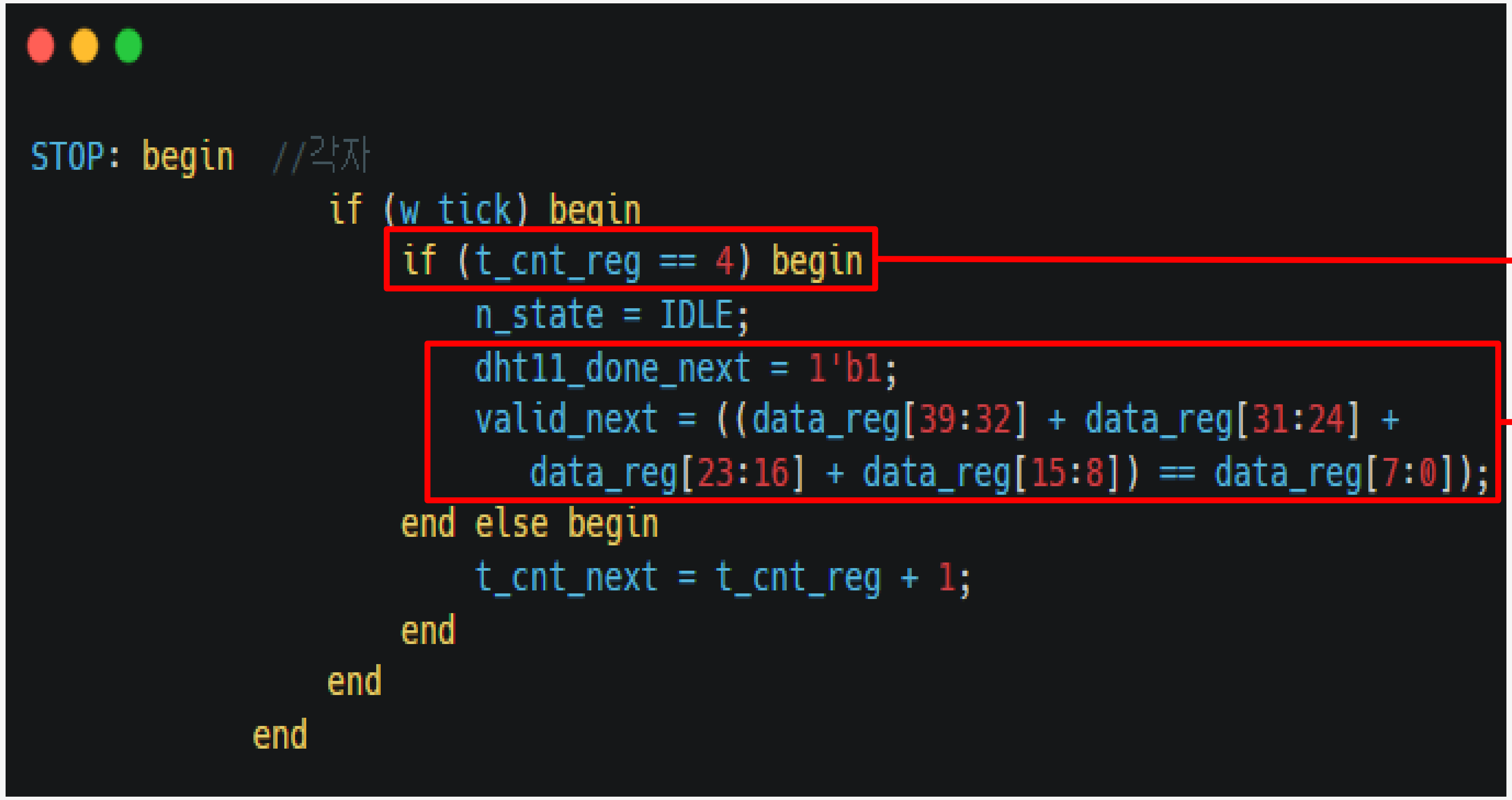
data in이 끝났을시 얼마나 지났나를 판별하여
값 shift 입력

데이터를 40번 읽었을시 STOP으로 이동
그 이하일시 DATA_SYNC로 이동하여 값 읽기

tick마다 cnt를 올려 H의 길이 측정

검증

code



```
STOP: begin  //각자
                if (w tick) begin
                    if (t_cnt_reg == 4) begin
                        n_state = IDLE;
                        dht11_done_next = 1'b1;
                        valid_next = ((data_reg[39:32] + data_reg[31:24] +
                           data_reg[23:16] + data_reg[15:8]) == data_reg[7:0]);
                    end else begin
                        t_cnt_next = t_cnt_reg + 1;
                    end
                end
            end
```

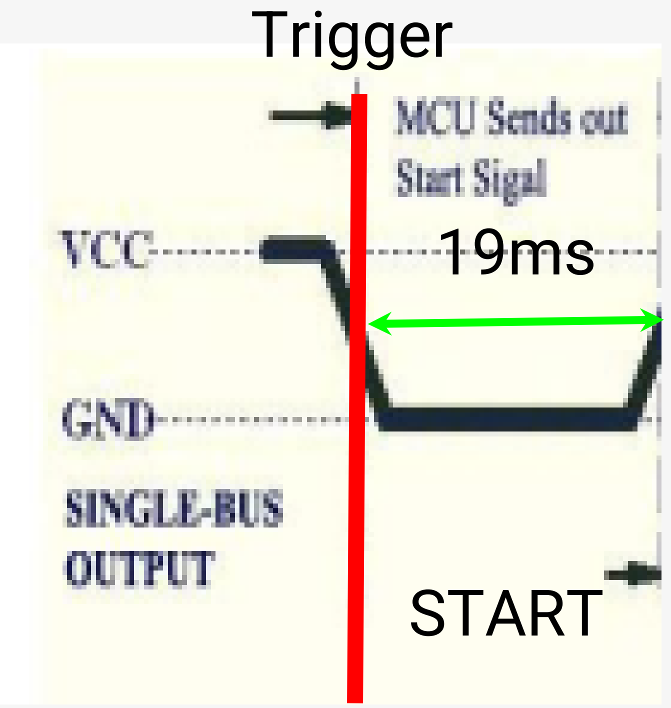
50us 대기

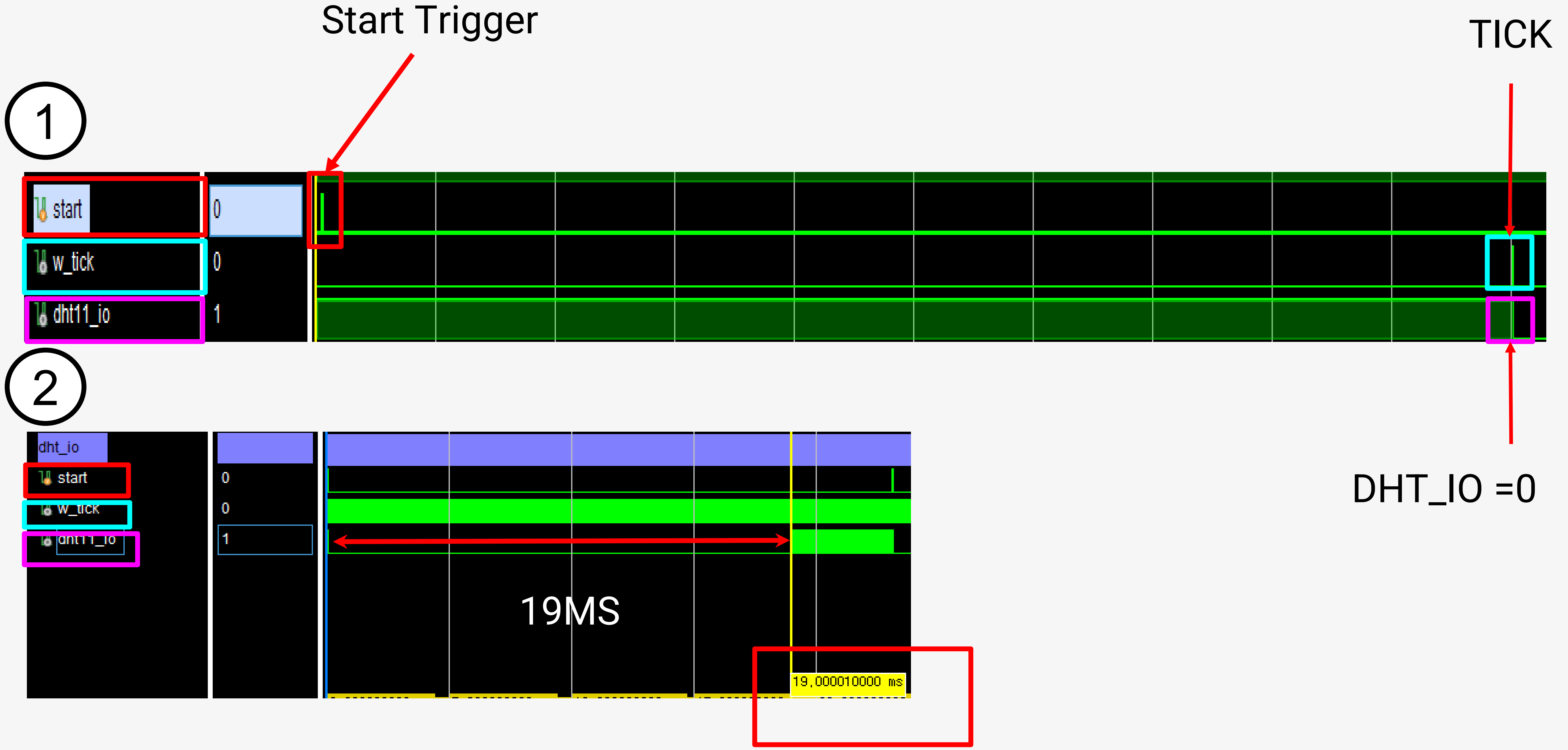
done과 valid(checksum)신호 출력

검증

protocol

START → DHT_IO 0 출력 (19ms)

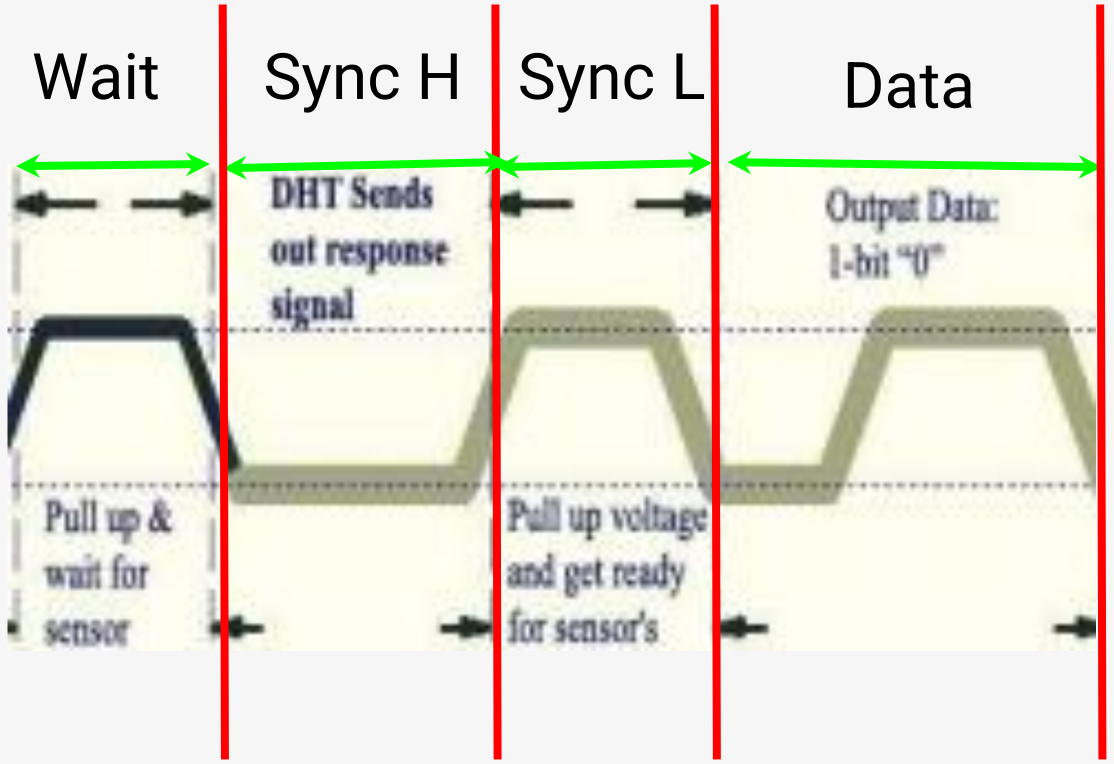


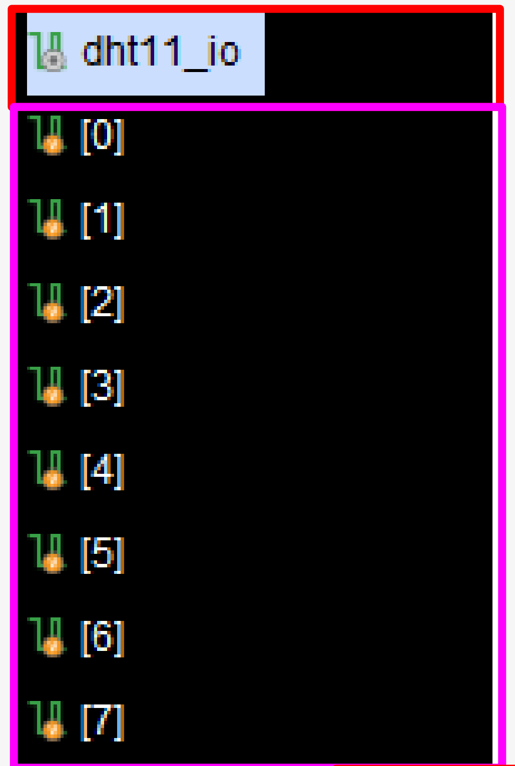


검증

protocol

Wait → Data Detect


Wait
Sync H
Sync L
Data
DHT Sends out response signal
Output Data: 1-bit "0"
Pull up & wait for sensor
Pull up voltage and get ready for sensor's

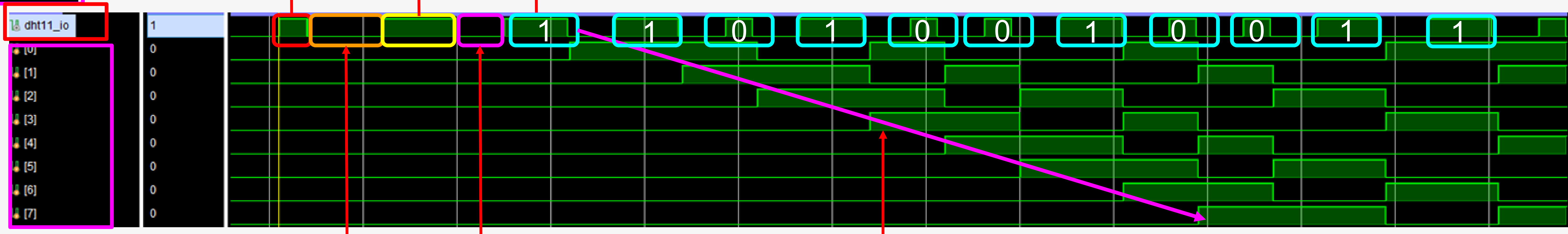


Wait

Sync L

Data

Repeat Catching Data Until 40bits



Data Shift Reg
40bit 중 8bit 일부

Sync H

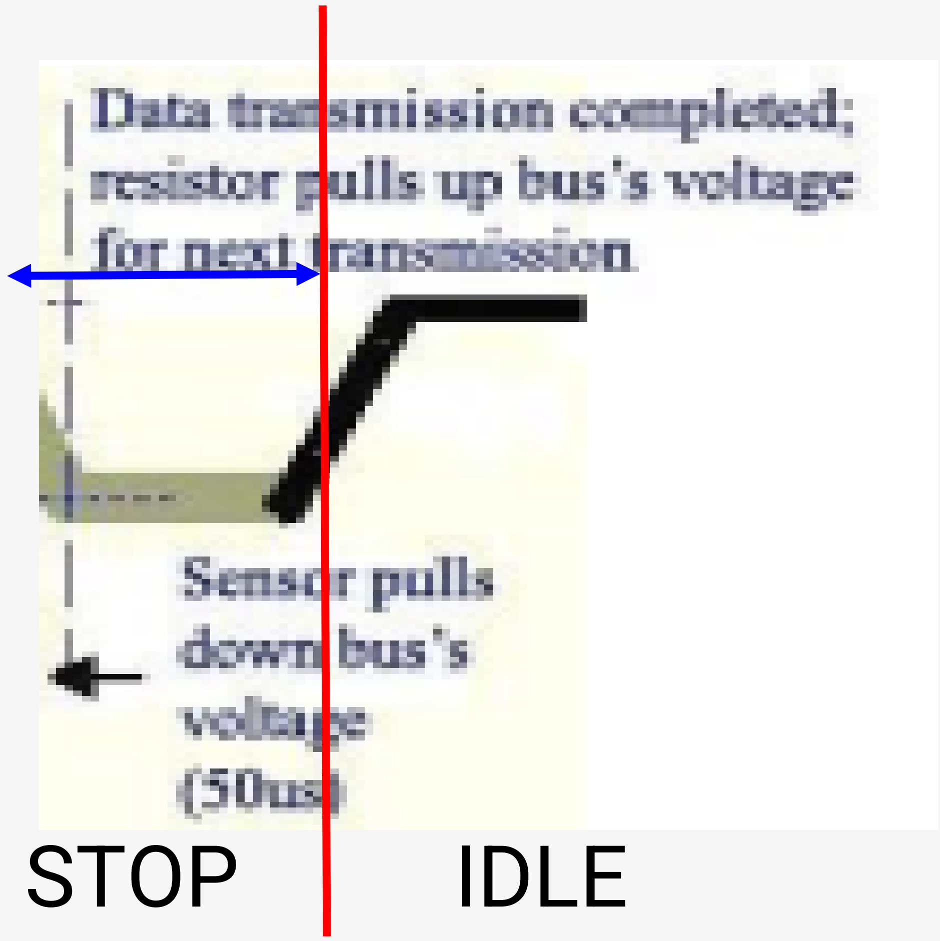
Data
Sync

Catch Data & Shift Data

검증

protocol

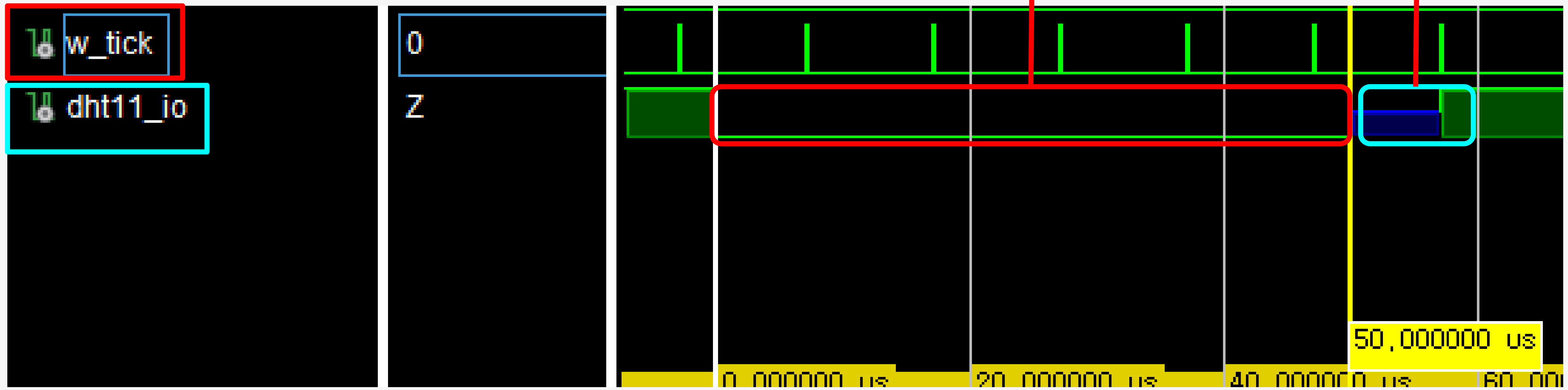
STOP


Data transmission completed; resistor pulls up bus's voltage for next transmission
Sensor pulls down bus's voltage (50us)

STOP IDLE

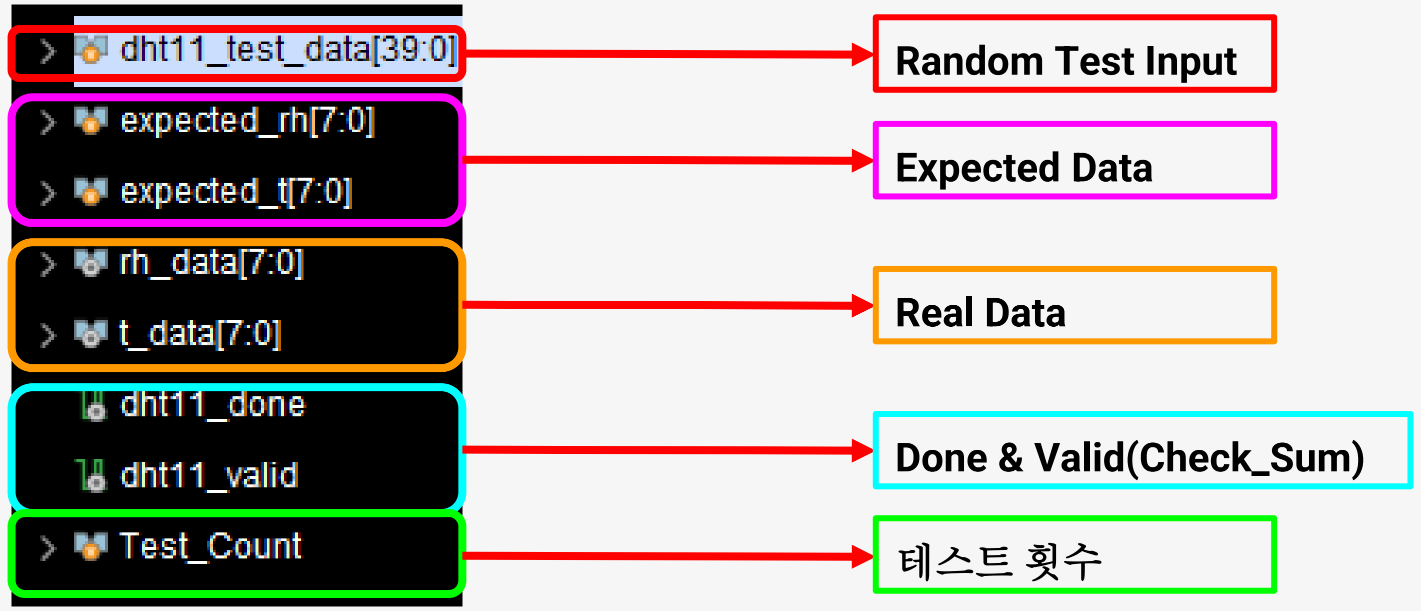
STOP: 50us

Z: Input → Output



검증

Function Random Test



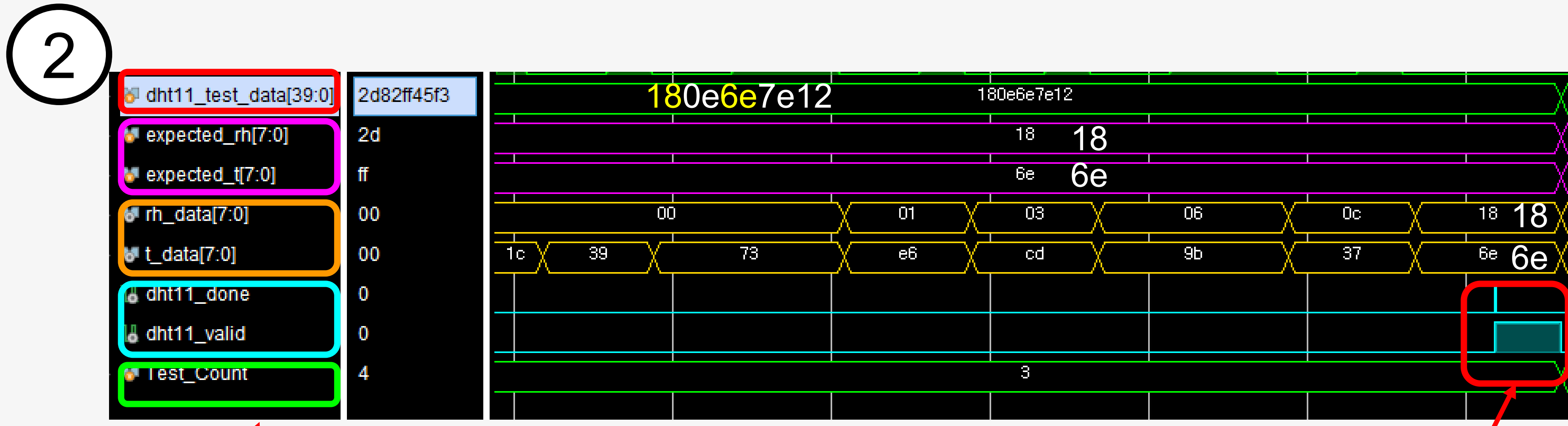
Random Test Input

Expected Data

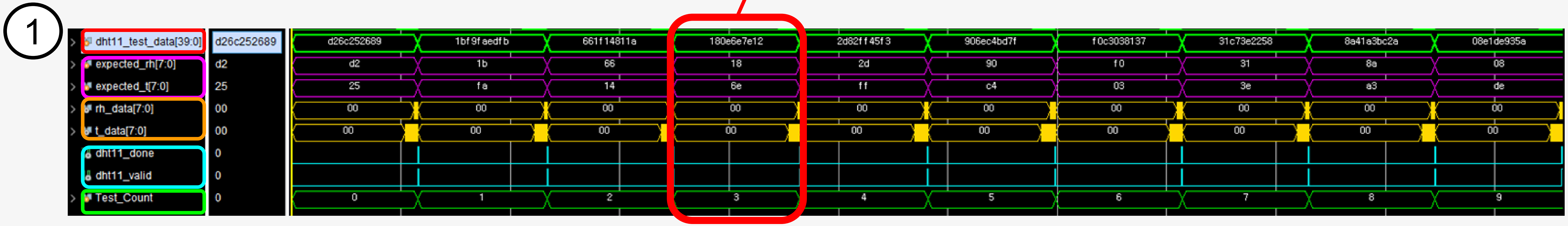
Real Data

Done & Valid(Check_Sum)

테스트 횟수

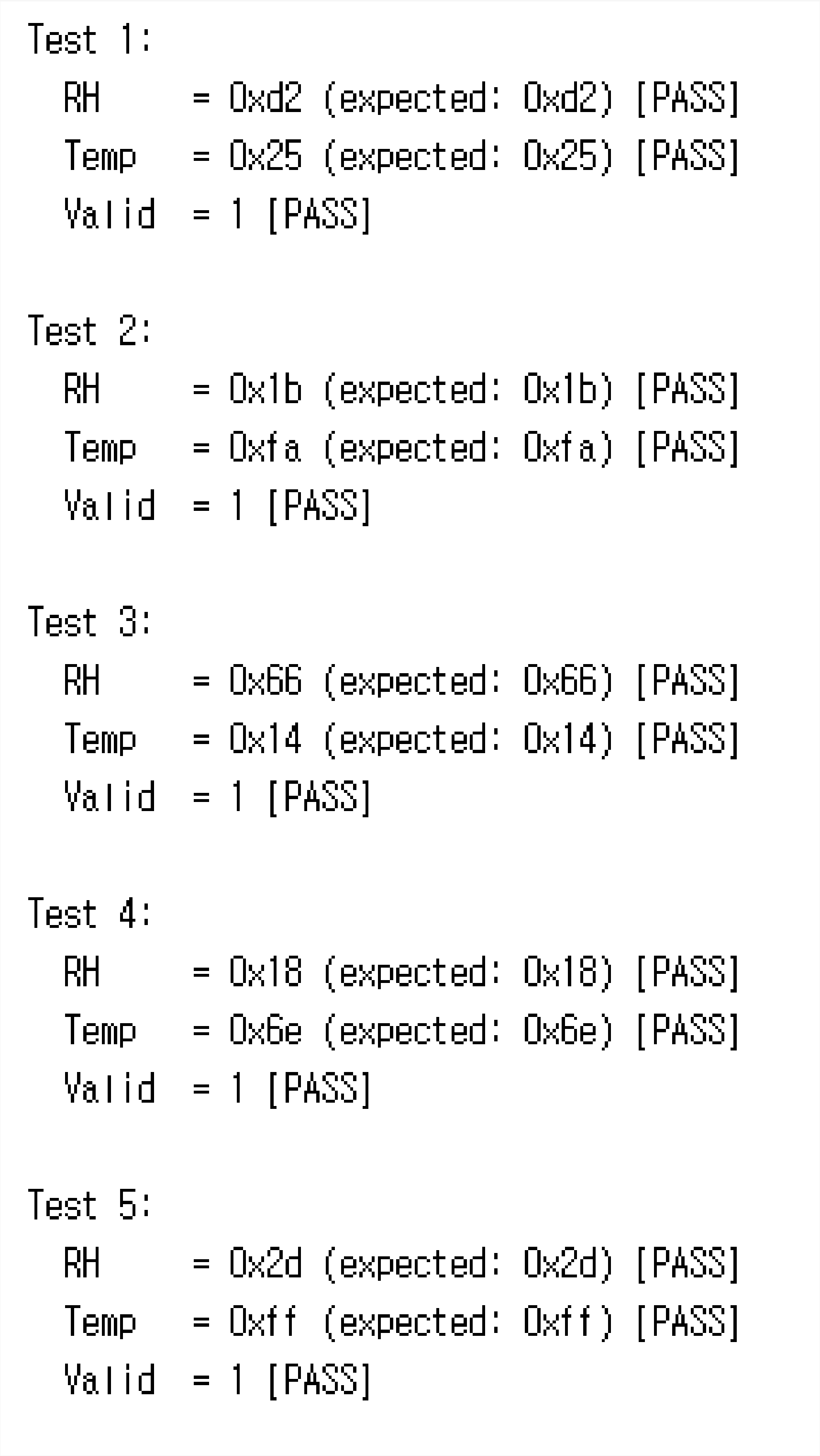


Done == 1
Valid == 1

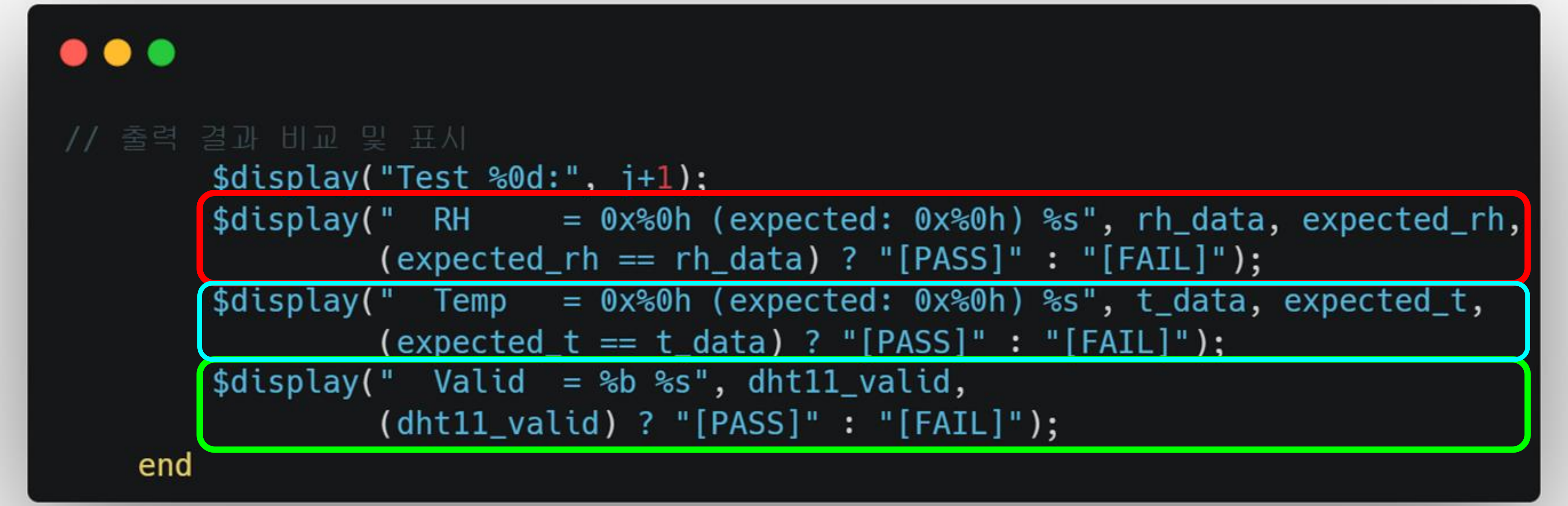

1
dht11_test_data[39:0] d26c252689
expected_rh[7:0] d2
expected_t[7:0] 25
rh_data[7:0] 00
t_data[7:0] 00
dht11_done 0
dht11_valid 0
Test_Count 0
d26c252689 1bf9faedfb 661f14811a 180e6e7e12 2d82ff45f3 906ec4bd7f f0c3038137 31c73e2258 8a41a3bc2a 08e1de935a
d2 1b 66 18 2d 90 f0 31 8a 08
25 fa 14 6e ff c4 03 3e a3 de
00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00
0 1 2 3 4 5 6 7 8 9

검증

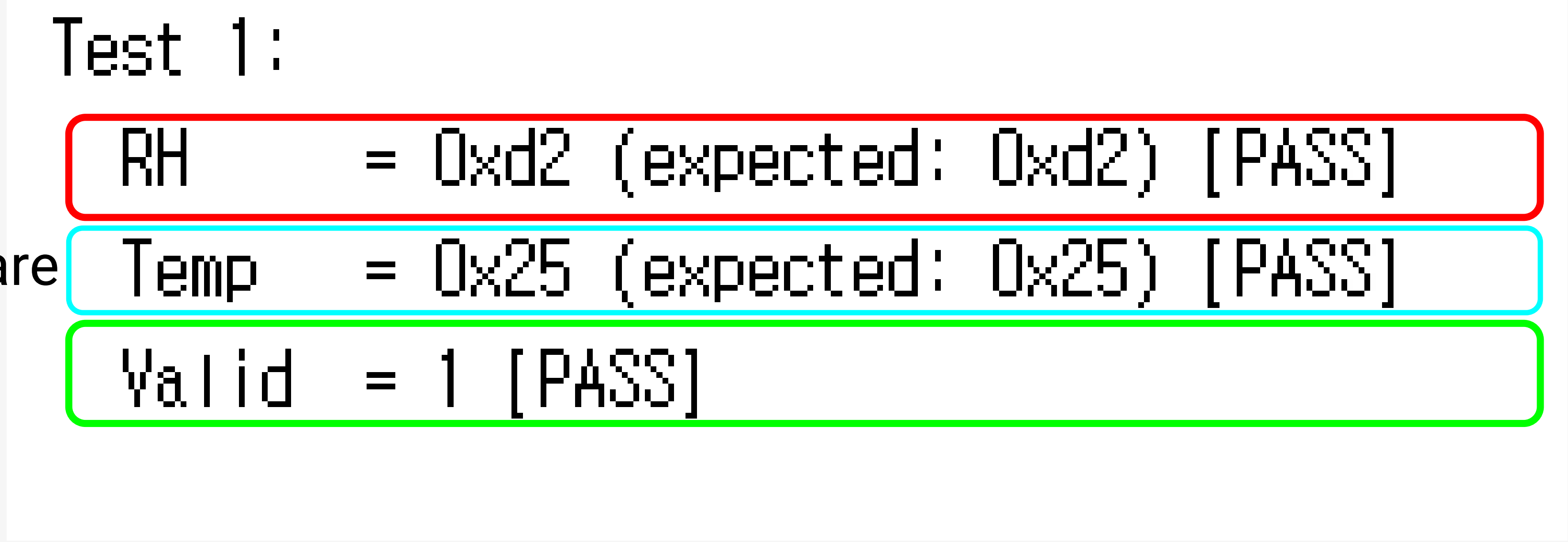
Function Random Test



```
Test 1:
  RH     = 0xd2 (expected: 0xd2) [PASS]
  Temp   = 0x25 (expected: 0x25) [PASS]
  Valid  = 1 [PASS]

Test 2:
  RH     = 0x1b (expected: 0x1b) [PASS]
  Temp   = 0xfa (expected: 0xfa) [PASS]
  Valid  = 1 [PASS]

Test 3:
  RH     = 0x66 (expected: 0x66) [PASS]
  Temp   = 0x14 (expected: 0x14) [PASS]
  Valid  = 1 [PASS]

Test 4:
  RH     = 0x18 (expected: 0x18) [PASS]
  Temp   = 0x6e (expected: 0x6e) [PASS]
  Valid  = 1 [PASS]

Test 5:
  RH     = 0x2d (expected: 0x2d) [PASS]
  Temp   = 0xff (expected: 0xff) [PASS]
  Valid  = 1 [PASS]
```



```
// 출력 결과 비교 및 표시
        $display("Test %0d:", i+1);
        $display("  RH     = 0x%0h (expected: 0x%0h) %s", rh_data, expected_rh,
                 (expected_rh == rh_data) ? "[PASS]" : "[FAIL]");
        $display("  Temp   = 0x%0h (expected: 0x%0h) %s", t_data, expected_t,
                 (expected_t == t_data) ? "[PASS]" : "[FAIL]");
        $display("  Valid  = %b %s", dht11_valid,
                 (dht11_valid) ? "[PASS]" : "[FAIL]");
    end
```



```
Test 1:
  RH     = 0xd2 (expected: 0xd2) [PASS]
  Temp   = 0x25 (expected: 0x25) [PASS]
  Valid  = 1 [PASS]
```

RH Compare

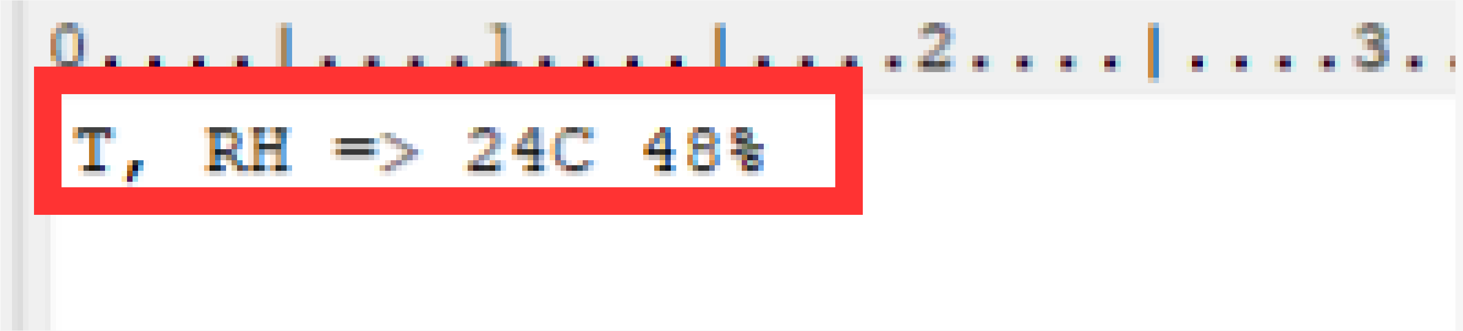
Temp Compare

Valid

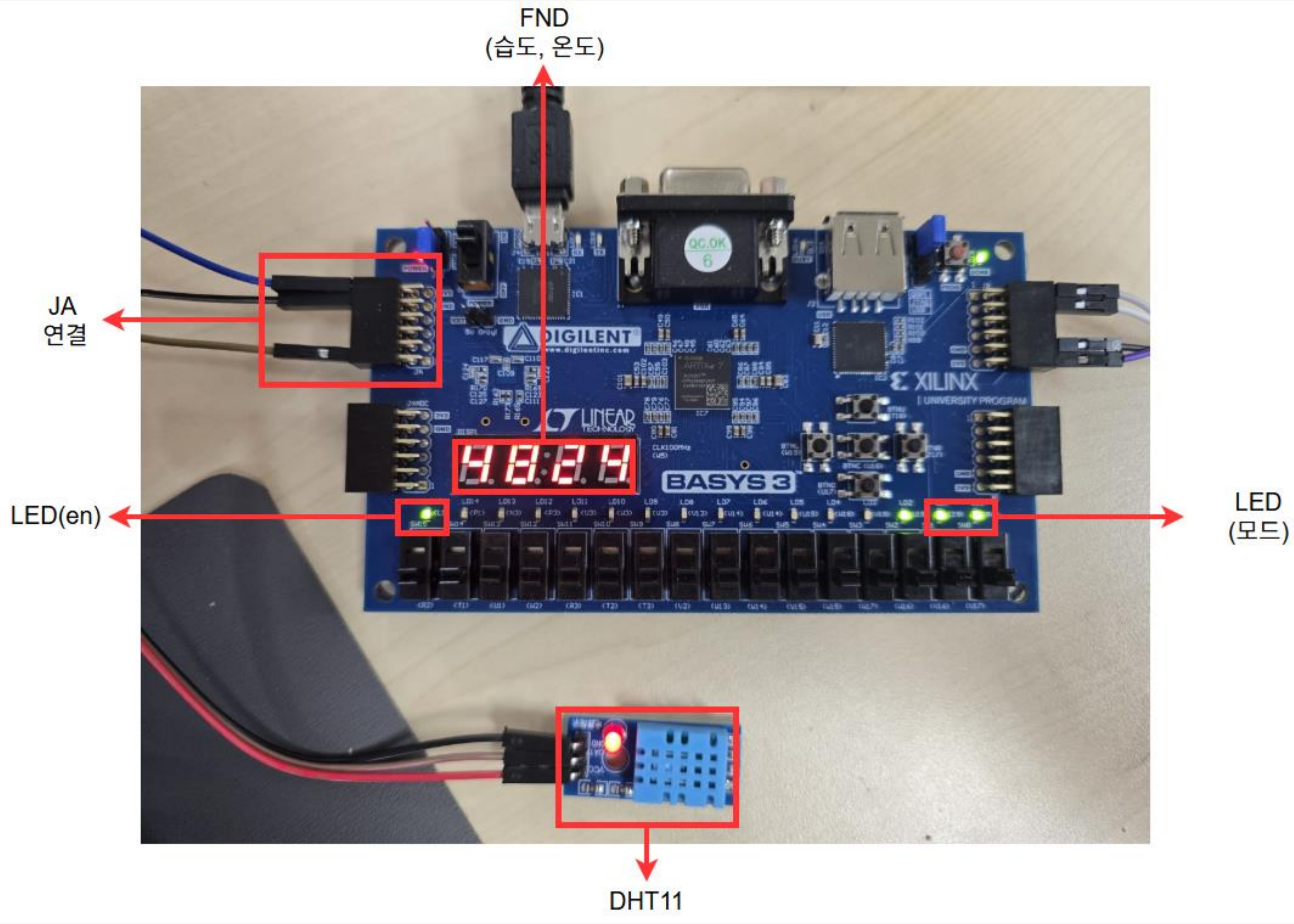
4. 결과

1. Problem
2. Solution

결과



```
0....|....1....|....2....|....3..
T, RH => 24C 48%
```



FND
(습도, 온도)

JA
연결

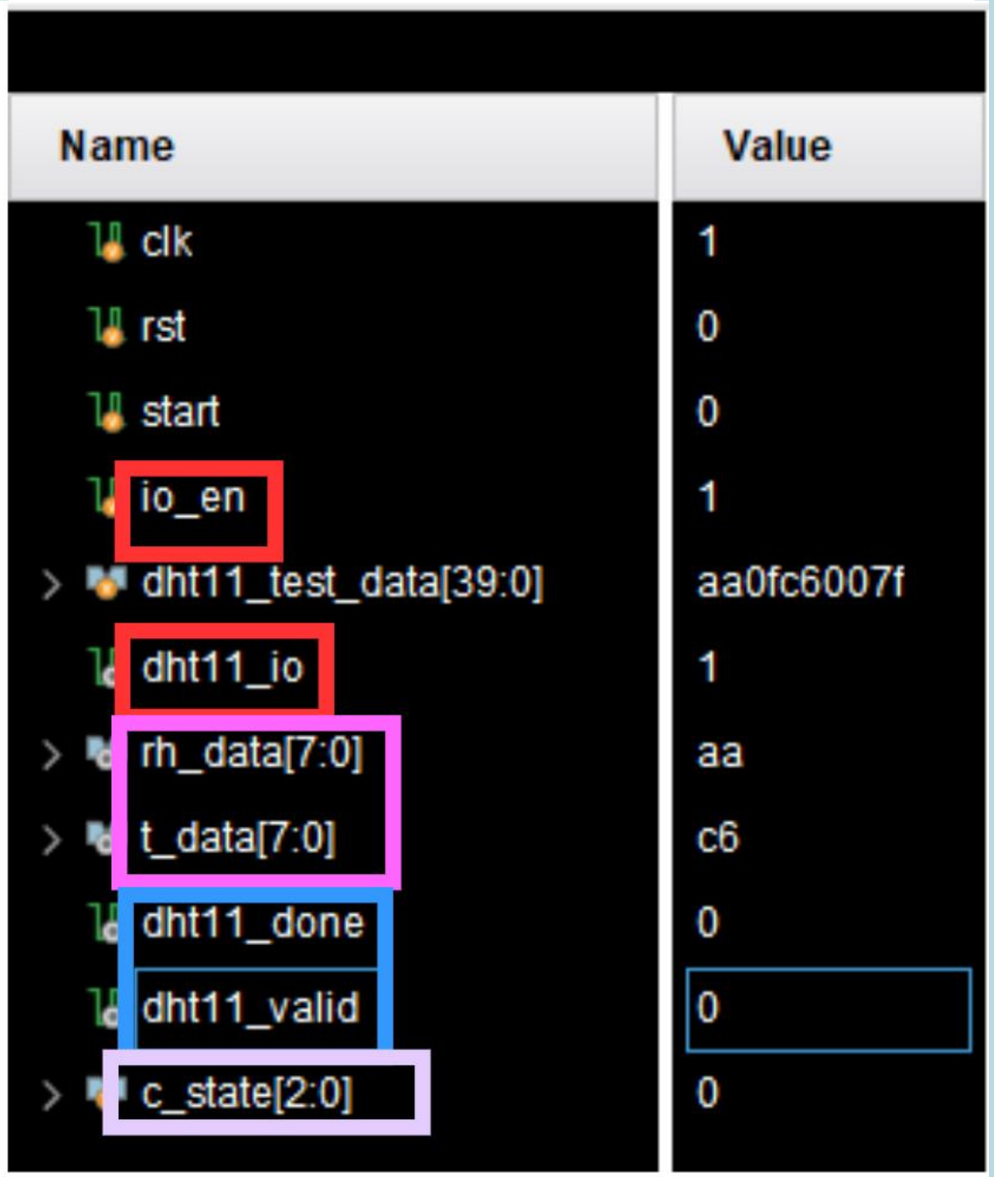
LED(en)

LED
(모드)

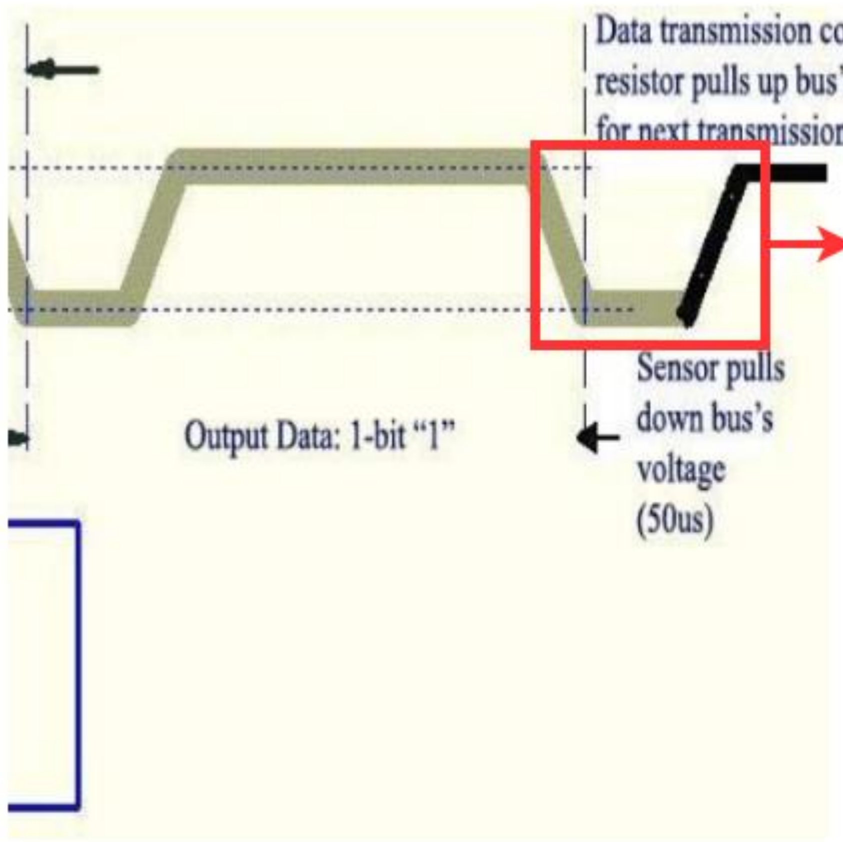
DHT11

5. 문제 해결

1. Problem
2. Solution



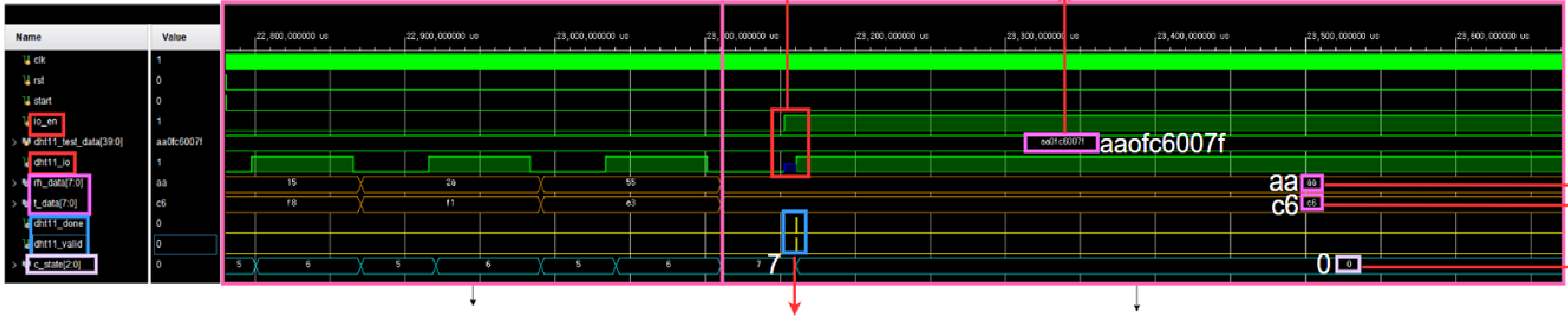
Name	Value
clk	1
rst	0
start	0
io_en	1
dht11_test_data[39:0]	aa0fc6007f
dht11_io	1
rh_data[7:0]	aa
t_data[7:0]	c6
dht11_done	0
dht11_valid	0
c_state[2:0]	0


Data transmission completed; resistor pulls up bus's voltage for next transmission
Output Data: 1-bit "1"
Sensor pulls down bus's voltage (50us)

50us low 이후 output으로 전환

보드에서 50us후 High 출력

입력한 값



aaofc6007f

aa → 습도 정수 부분 정상 출력

c6 → 온도 정수 부분 정상 출력

7

0 → IDLE state

DATA_DETECT까지 정상 작동

done과 valid 정상 출력

DATA_DETECT까지 정상 작동

Merge

1. 개요
2. 상세설계
3. 검증
4. 결과
5. 문제해결
6. 고찰

개요

1.spec

-Board

-HC-SR04

-DHT11

BOARD SPEC

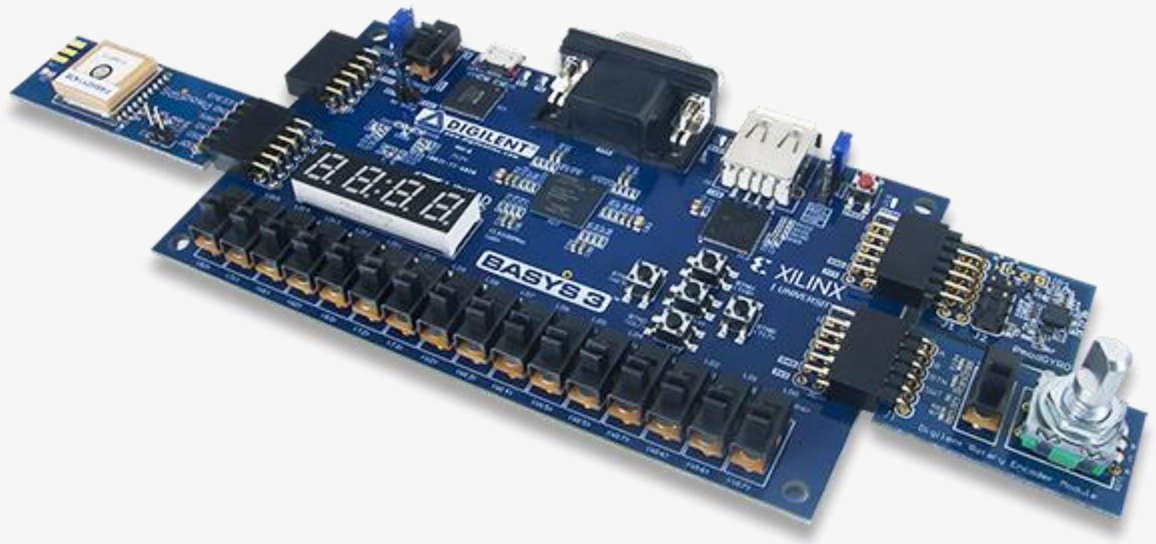
-16 users switches

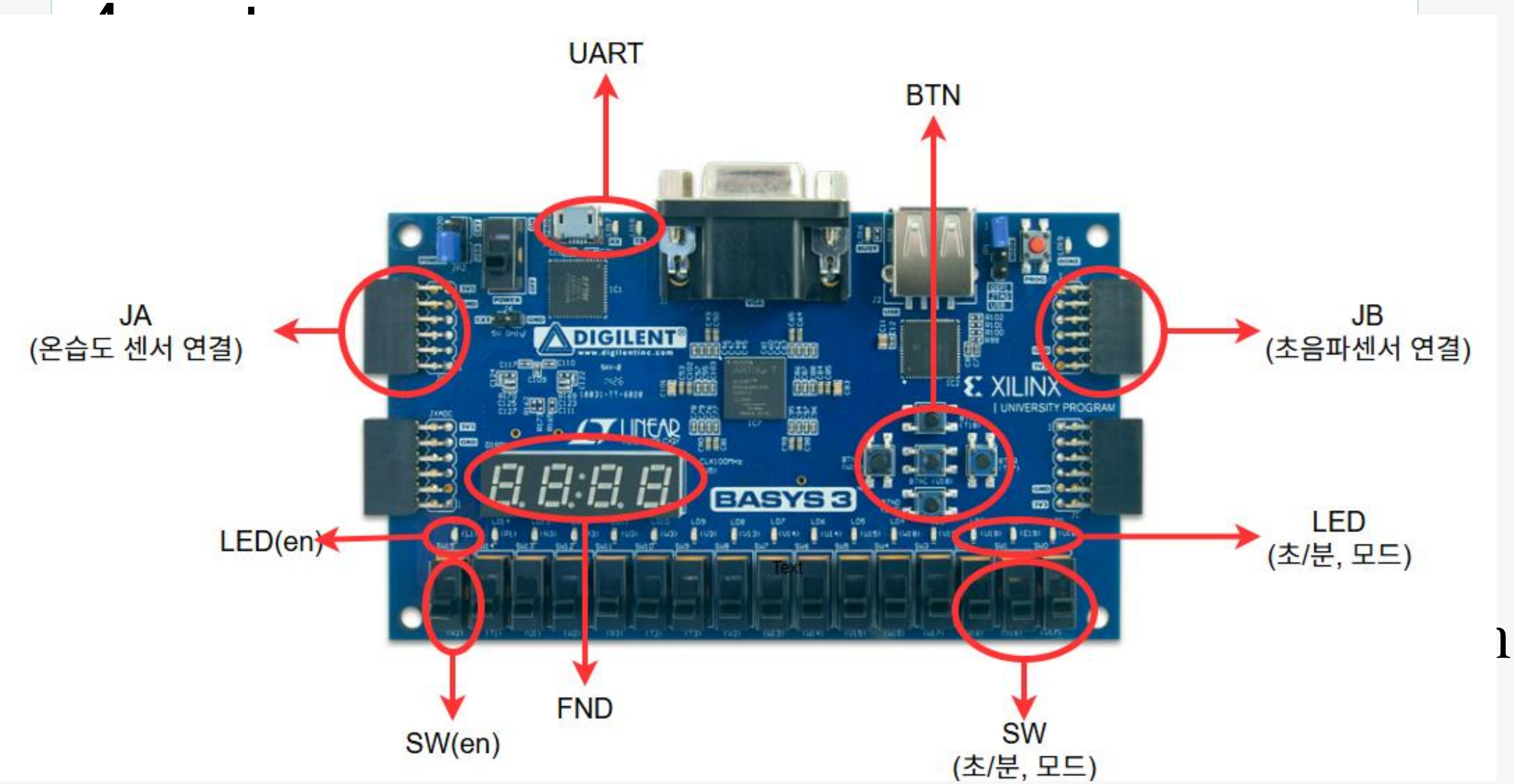
-16 users leds

-5 users push buttons

-4 digit 7 segment display

-3.3v logic level





개요

1.spec

-Board 입력

UART

M: SW[0] 토글

N: SW[1] 토글

E: en 활성화

W: 초/분 변경

U: UP BUTTON

D: DOWN BUTTON

L: LEFT BUTTON

R: RIGHT BUTTON

실제 입력

SW[1:0]: mode 변경

SW[2]: 초/분 변경

SW[3]: en

LED[1:0]: mode 표시

LED[2]: 초/분

LED[3]: en

개요

1.spec

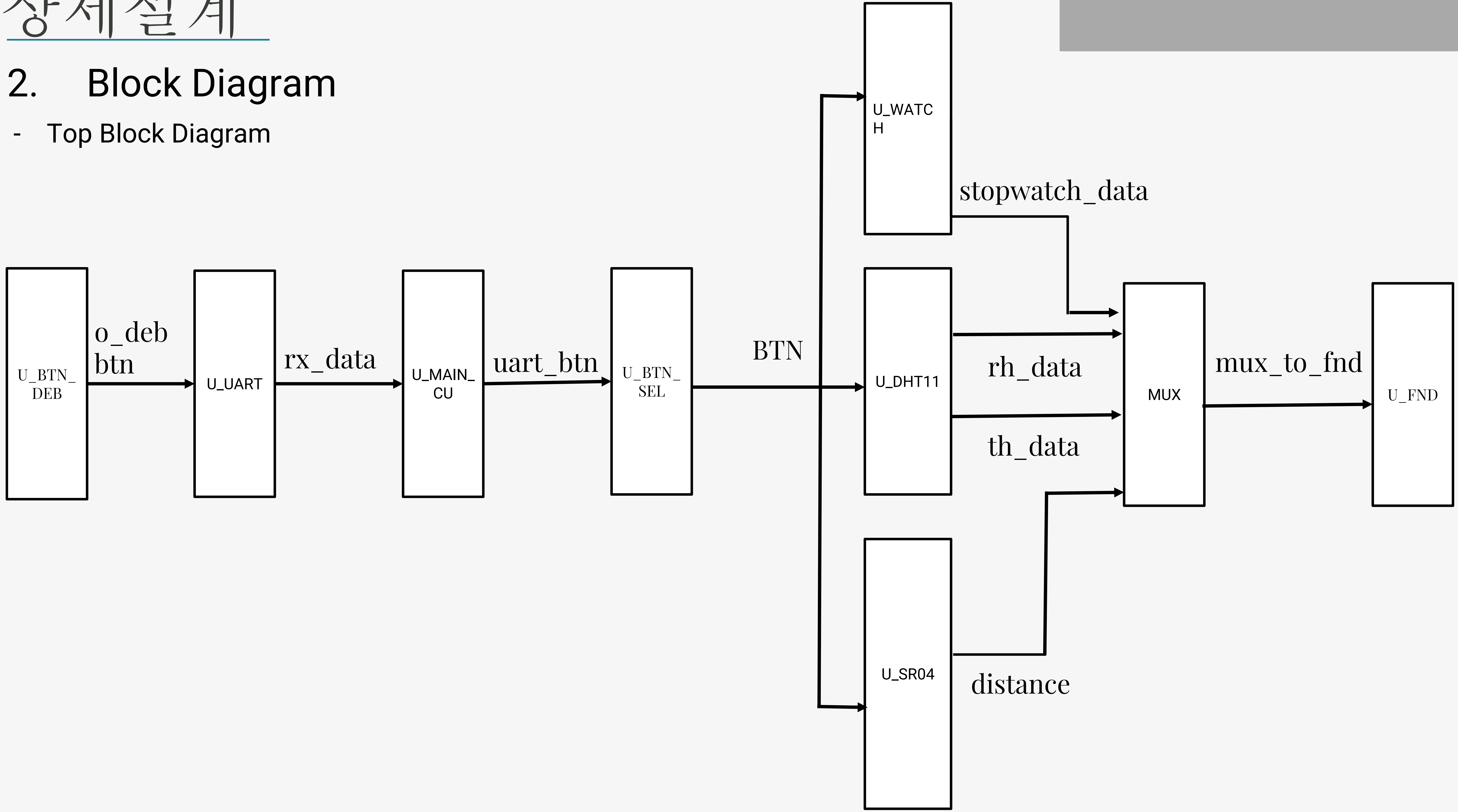
-Board btn 입력

	watch	stopwatch	초음파 센서	온습도 센서
U	값 올림		측정	측정
D	값 내림			
L	선택 왼쪽(큰쪽) 이동			
R	선택 오른쪽(작은쪽) 이동			
G		시작		
S		정지		
C		clear		
ESC	reset	reset	reset	reset

상세설계

2. Block Diagram

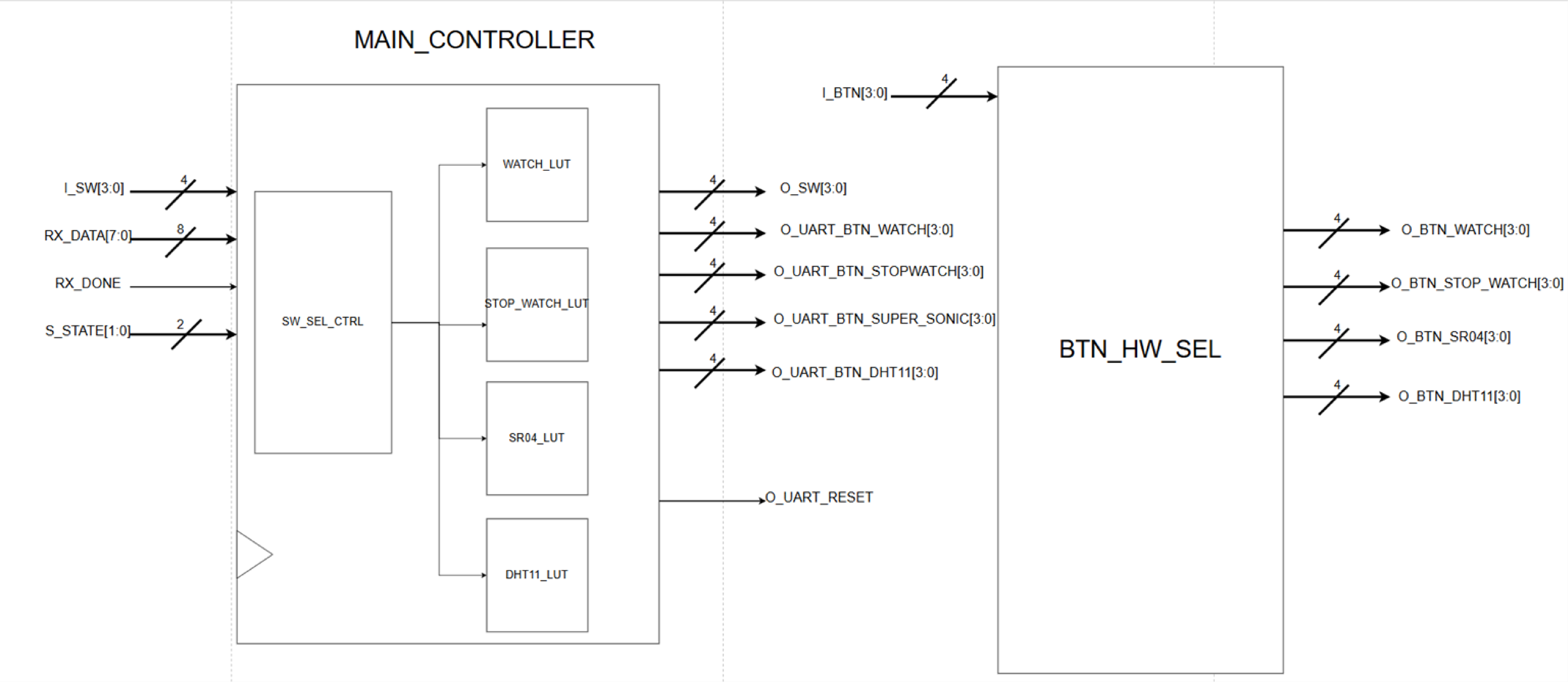
- Top Block Diagram



상세설계

2. Block Diagram

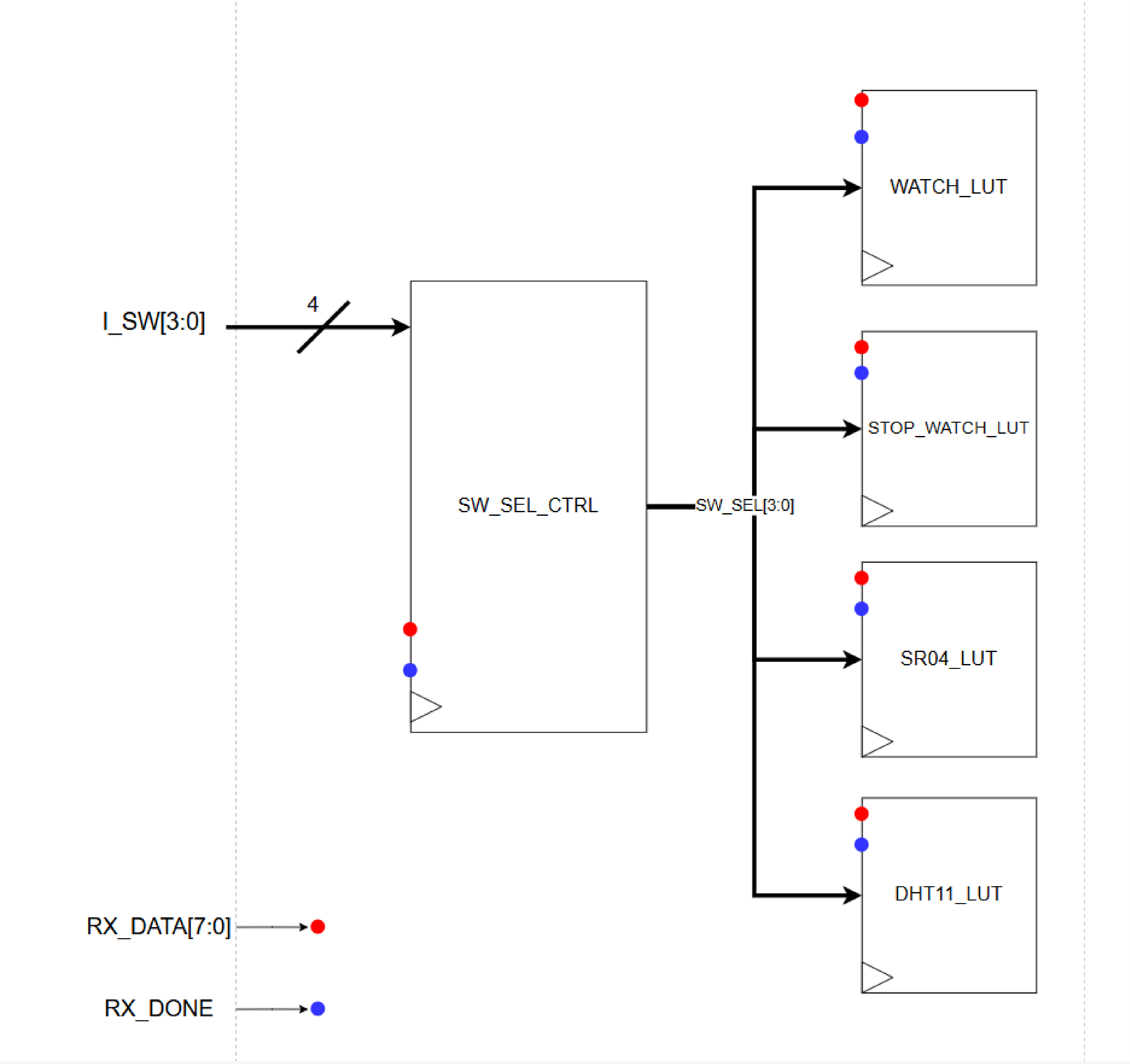
- Main Controller Block Diagram



상세설계

2. Detail HW Architecture

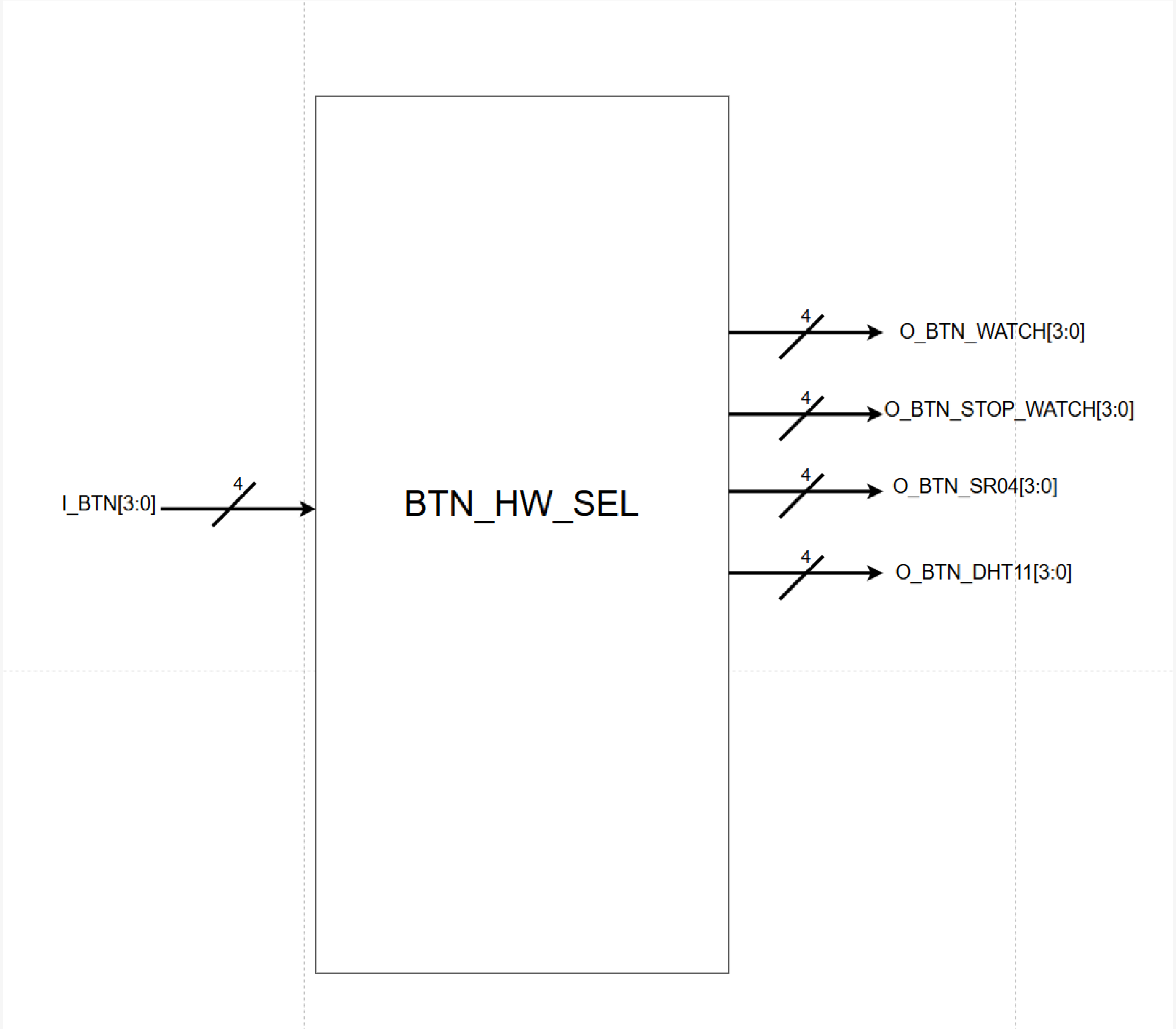
- SW_SEL_CTRL & LUT



상세설계

2. Full HW Architecture

- BTN_HW_SEL

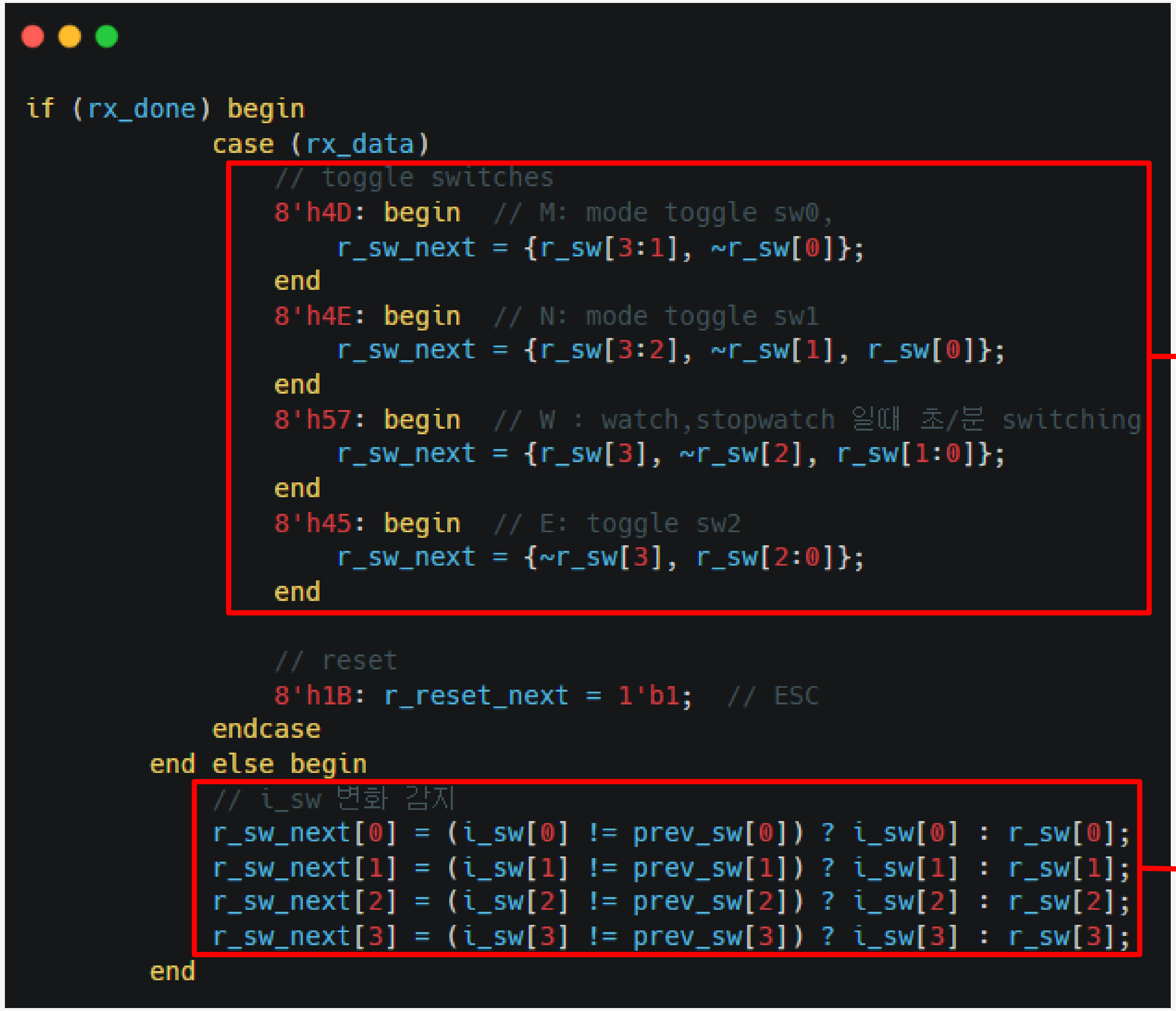


3.검증

1. Code
2. Simulation

검증

code: sw select



```
if (rx_done) begin
      case (rx_data)
        // toggle switches
        8'h4D: begin  // M: mode toggle sw0,
          r_sw_next = {r_sw[3:1], ~r_sw[0]};
        end
        8'h4E: begin  // N: mode toggle sw1
          r_sw_next = {r_sw[3:2], ~r_sw[1], r_sw[0]};
        end
        8'h57: begin  // W : watch,stopwatch 일때 초/분 switching
          r_sw_next = {r_sw[3], ~r_sw[2], r_sw[1:0]};
        end
        8'h45: begin  // E: toggle sw2
          r_sw_next = {~r_sw[3], r_sw[2:0]};
        end

        // reset
        8'h1B: r_reset_next = 1'b1;  // ESC
      endcase
    end else begin
      // i_sw 변화 감지
      r_sw_next[0] = (i_sw[0] != prev_sw[0]) ? i_sw[0] : r_sw[0];
      r_sw_next[1] = (i_sw[1] != prev_sw[1]) ? i_sw[1] : r_sw[1];
      r_sw_next[2] = (i_sw[2] != prev_sw[2]) ? i_sw[2] : r_sw[2];
      r_sw_next[3] = (i_sw[3] != prev_sw[3]) ? i_sw[3] : r_sw[3];
    end
```

uart로 들어온 입력을 통해 해당 부분만 toggle

물리 sw입력시, 이전과 달라졌다면 해당 부분만 toggle

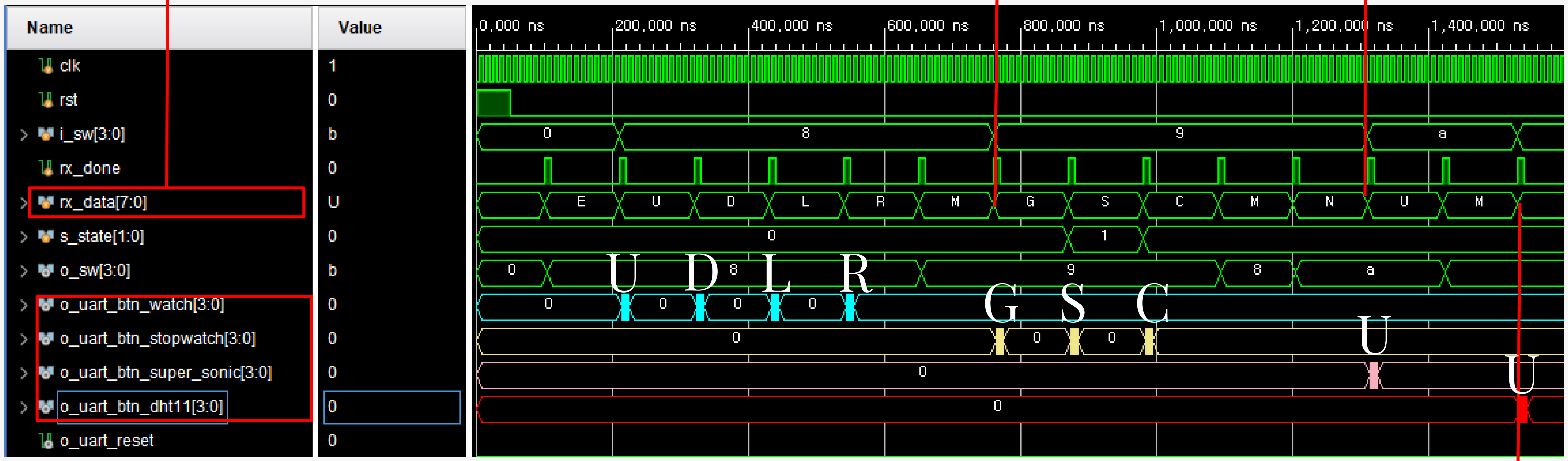
검증

2. Full HW Architecture

- BTN_HW_SEL

uart입력

w/t에서s/w로 전환s/w에서 sonic으로 전환

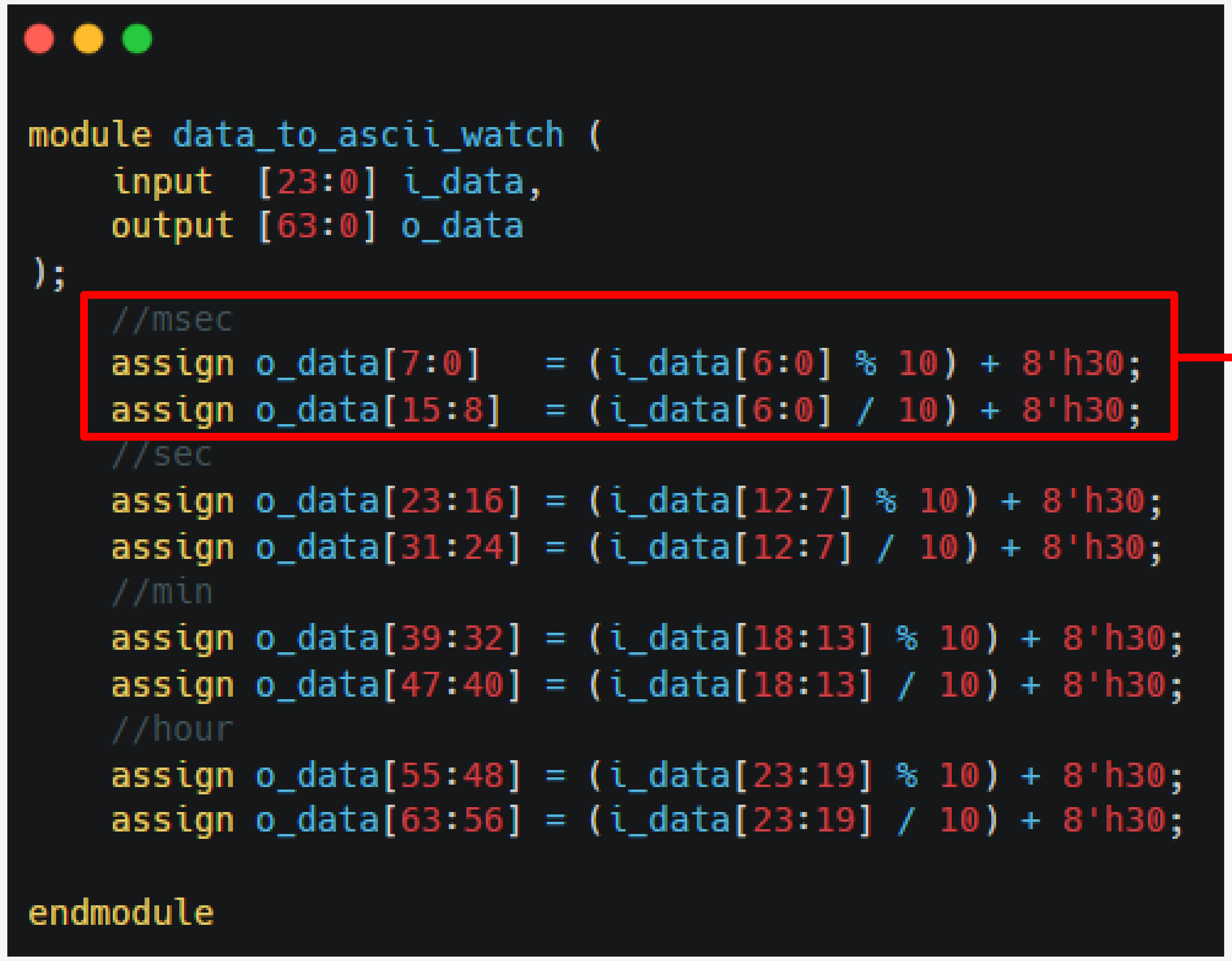


Name	Value
clk	1
rst	0
i_sw[3:0]	b
rx_done	0
rx_data[7:0]	U
s_state[1:0]	0
o_sw[3:0]	b
o_uart_btn_watch[3:0]	0
o_uart_btn_stopwatch[3:0]	0
o_uart_btn_super_sonic[3:0]	0
o_uart_btn_dht11[3:0]	0
o_uart_reset	0

sro4에서 dht로전환

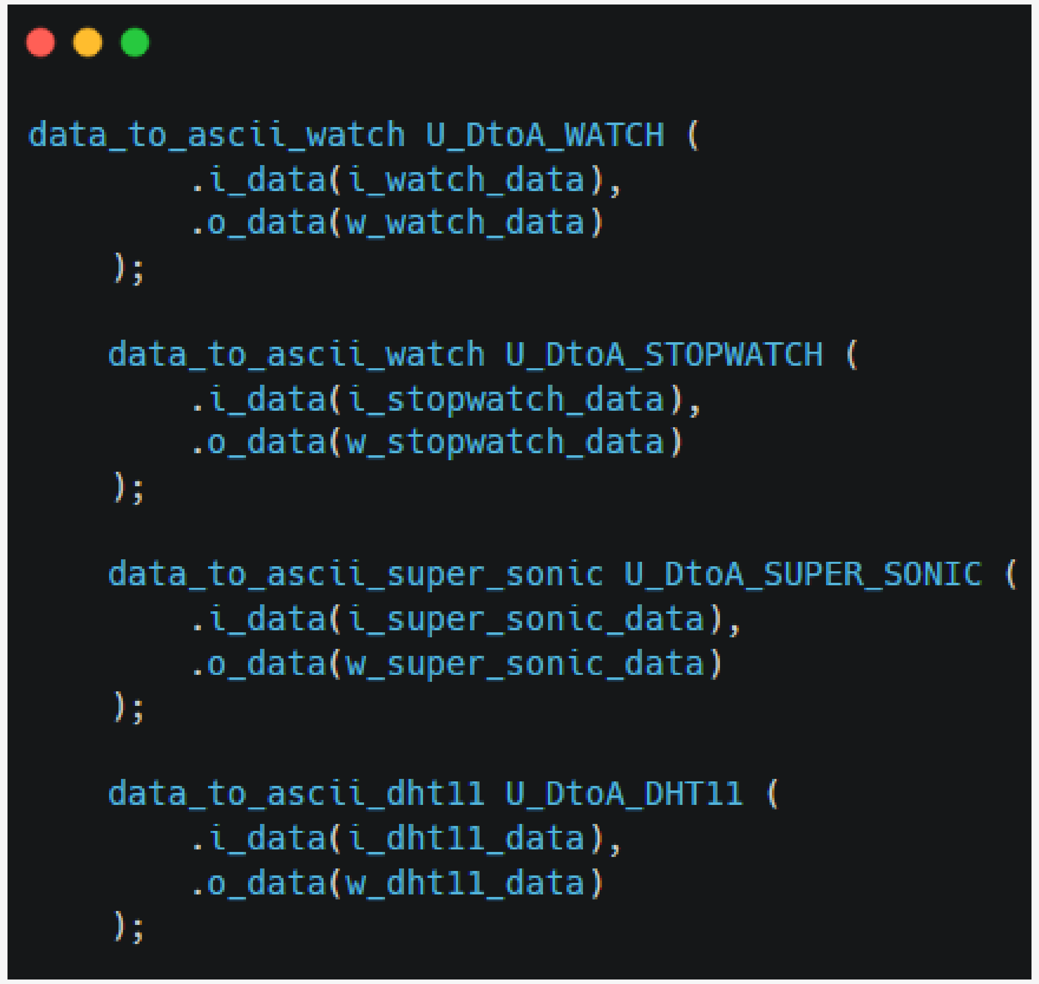
검증

code: ascii 변환



```
module data_to_ascii_watch (
    input  [23:0] i_data,
    output [63:0] o_data
);
    //msec
    assign o_data[7:0]    = (i_data[6:0] % 10) + 8'h30;
    assign o_data[15:8]   = (i_data[6:0] / 10) + 8'h30;
    //sec
    assign o_data[23:16] = (i_data[12:7] % 10) + 8'h30;
    assign o_data[31:24] = (i_data[12:7] / 10) + 8'h30;
    //min
    assign o_data[39:32] = (i_data[18:13] % 10) + 8'h30;
    assign o_data[47:40] = (i_data[18:13] / 10) + 8'h30;
    //hour
    assign o_data[55:48] = (i_data[23:19] % 10) + 8'h30;
    assign o_data[63:56] = (i_data[23:19] / 10) + 8'h30;

endmodule
```

data를 ascii로 변환

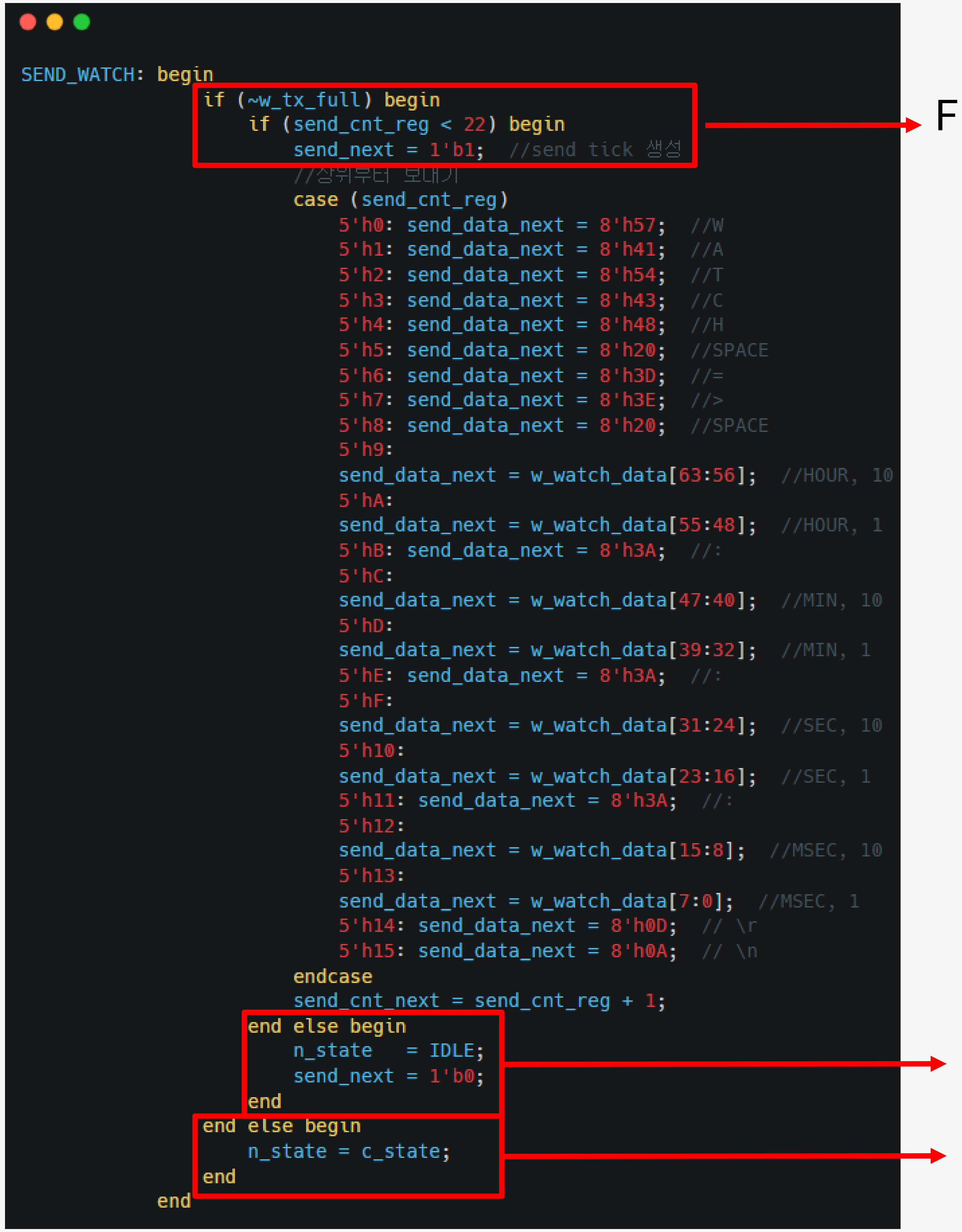


```
data_to_ascii_watch U_DtoA_WATCH (
        .i_data(i_watch_data),
        .o_data(w_watch_data)
    );

    data_to_ascii_watch U_DtoA_STOPWATCH (
        .i_data(i_stopwatch_data),
        .o_data(w_stopwatch_data)
    );

    data_to_ascii_super_sonic U_DtoA_SUPER_SONIC (
        .i_data(i_super_sonic_data),
        .o_data(w_super_sonic_data)
    );

    data_to_ascii_dht11 U_DtoA_DHT11 (
        .i_data(i_dht11_data),
        .o_data(w_dht11_data)
    );
```

인스턴스화

검증

code: FIFO로 IN



```
SEND_WATCH: begin
                if (~w_tx_full) begin
                    if (send_cnt_reg < 22) begin
                        send_next = 1'b1;  //send tick 생성
                        //상위부터 보내기
                        case (send_cnt_reg)
                            5'h0: send_data_next = 8'h57;  //W
                            5'h1: send_data_next = 8'h41;  //A
                            5'h2: send_data_next = 8'h54;  //T
                            5'h3: send_data_next = 8'h43;  //C
                            5'h4: send_data_next = 8'h48;  //H
                            5'h5: send_data_next = 8'h20;  //SPACE
                            5'h6: send_data_next = 8'h3D;  //=
                            5'h7: send_data_next = 8'h3E;  //>
                            5'h8: send_data_next = 8'h20;  //SPACE
                            5'h9:
                            send_data_next = w_watch_data[63:56];  //HOUR, 10
                            5'hA:
                            send_data_next = w_watch_data[55:48];  //HOUR, 1
                            5'hB: send_data_next = 8'h3A;  //:
                            5'hC:
                            send_data_next = w_watch_data[47:40];  //MIN, 10
                            5'hD:
                            send_data_next = w_watch_data[39:32];  //MIN, 1
                            5'hE: send_data_next = 8'h3A;  //:
                            5'hF:
                            send_data_next = w_watch_data[31:24];  //SEC, 10
                            5'h10:
                            send_data_next = w_watch_data[23:16];  //SEC, 1
                            5'h11: send_data_next = 8'h3A;  //:
                            5'h12:
                            send_data_next = w_watch_data[15:8];  //MSEC, 10
                            5'h13:
                            send_data_next = w_watch_data[7:0];  //MSEC, 1
                            5'h14: send_data_next = 8'h0D;  // \r
                            5'h15: send_data_next = 8'h0A;  // \n
                        endcase
                        send_cnt_next = send_cnt_reg + 1;
                    end else begin
                        n_state   = IDLE;
                        send_next = 1'b0;
                    end
                end else begin
                    n_state = c_state;
                end
            end
```

FIFO가 full이 아닐시, 순서대로 글자 입력

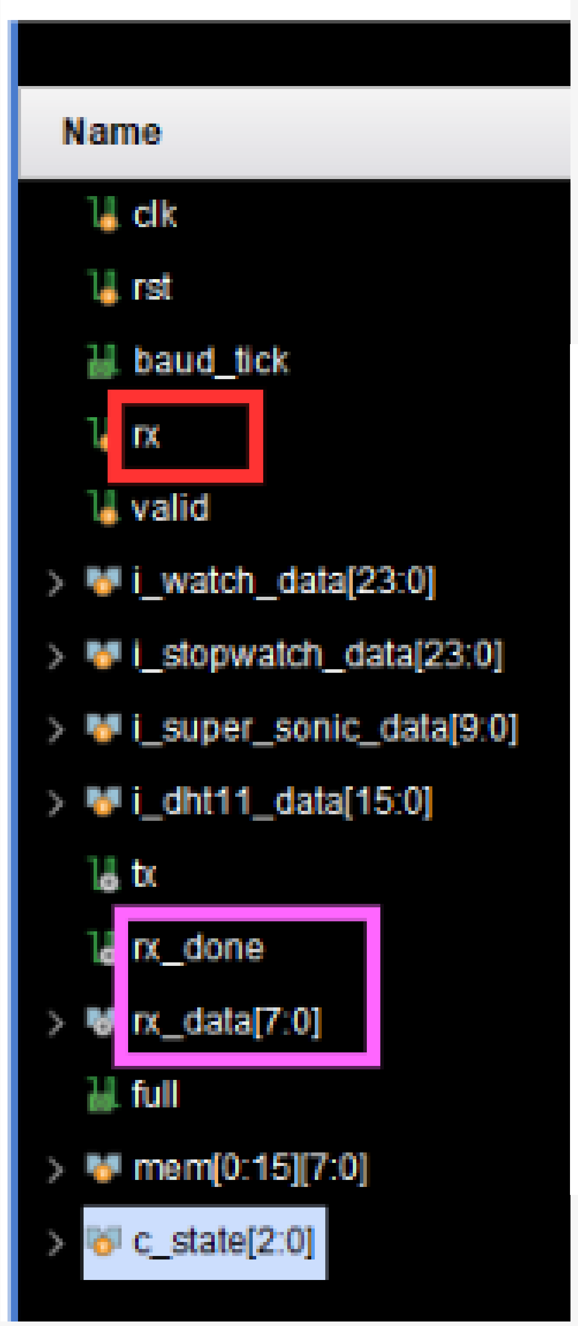
다 보냈을시, IDLE로 이동

full일시 현재 state에서 대기

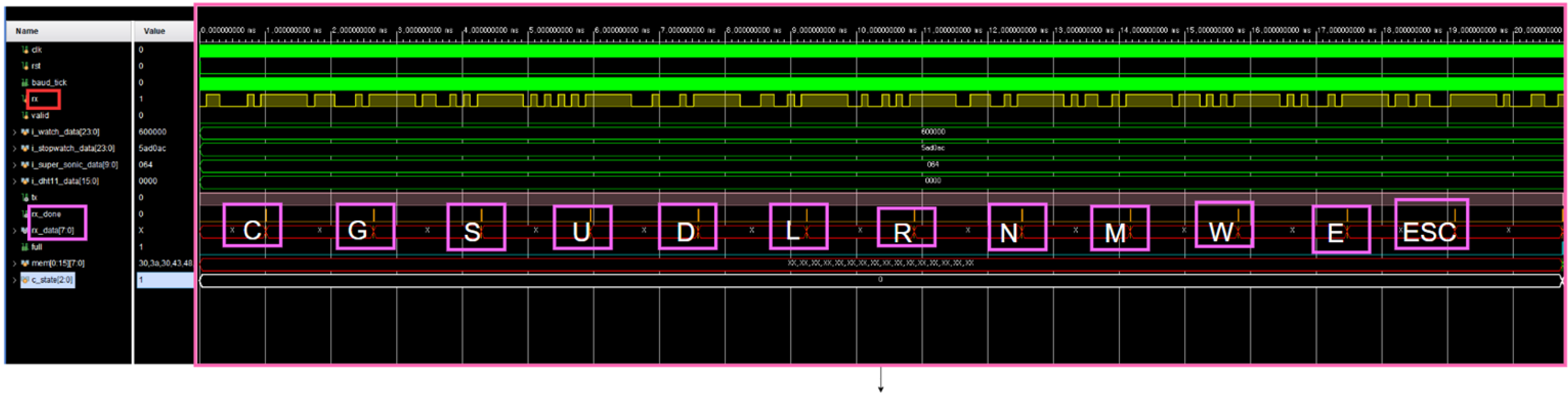
검증

2. UART

- rx로 명령어 입력



Name
clk
rst
baud_tick
rx
valid
i_watch_data[23:0]
i_stopwatch_data[23:0]
i_super_sonic_data[9:0]
i_dht11_data[15:0]
tx
rx_done
rx_data[7:0]
full
mem[0:15][7:0]
c_state[2:0]



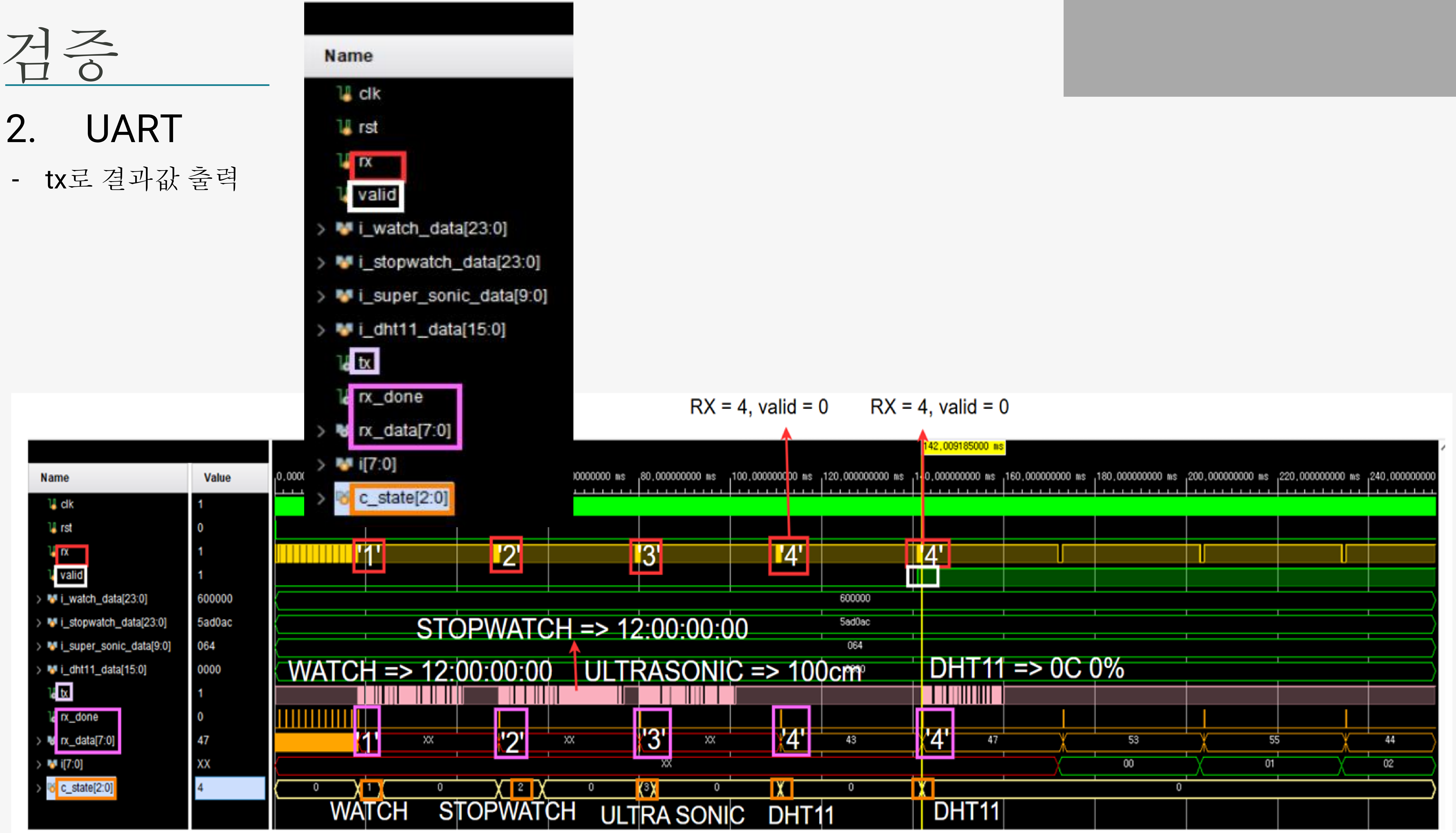
Name	Value
clk	0
rst	0
baud_tick	0
rx	1
valid	0
i_watch_data[23:0]	600000
i_stopwatch_data[23:0]	5ad0ac
i_super_sonic_data[9:0]	064
i_dht11_data[15:0]	0000
tx	0
rx_done	0
rx_data[7:0]	X
full	1
mem[0:15][7:0]	30,3a,30,43,48
c_state[2:0]	1

컨트롤 신호 입력

검증

2. UART

- tx로 결과값 출력



RX = 4, valid = 0

RX = 4, valid = 0

STOPWATCH => 12:00:00:00

WATCH => 12:00:00:00

ULTRASONIC => 100cm

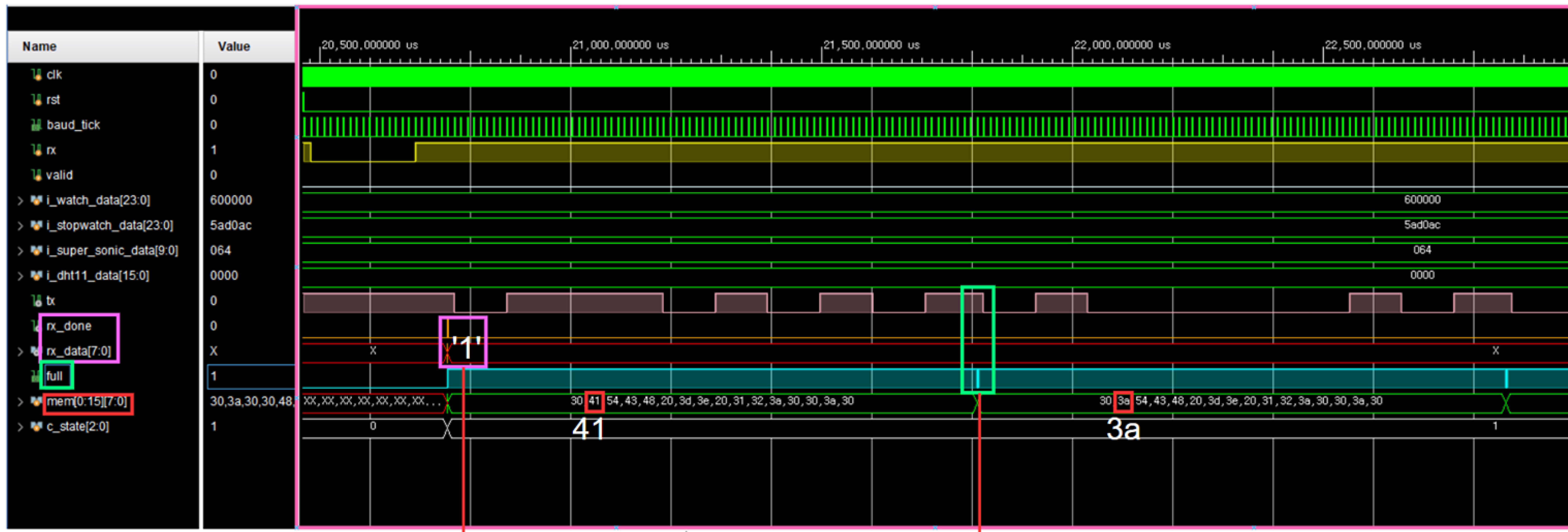
DHT11 => 0C 0%

WATCH STOPWATCH ULTRA SONIC DHT11 DHT11

검증

2. UART

- full시 FIFO 입력 대기



Name	Value
clk	0
rst	0
baud_tick	0
rx	1
valid	0
i_watch_data[23:0]	600000
i_stopwatch_data[23:0]	5ad0ac
i_super_sonic_data[9:0]	064
i_dht11_data[15:0]	0000
tx	0
rx_done	0
rx_data[7:0]	X
full	1
mem[0:15][7:0]	30,3a,30,30,48
c_state[2:0]	1

'1'입력

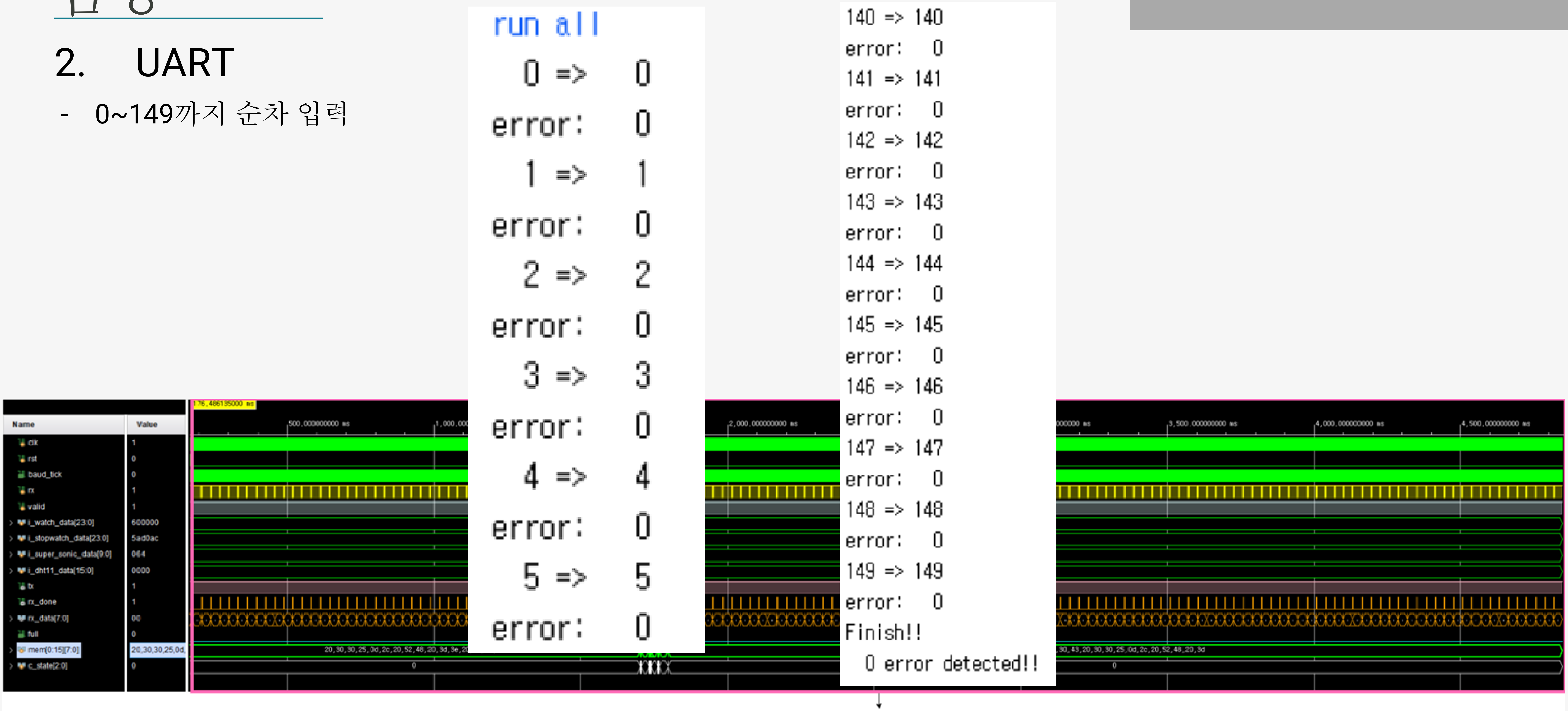
full시 fifo입력 대기

한 문자 전송 후 fifo에 입력

검증

2. UART

- 0~149까지 순차 입력



```
run all
  0 =>   0
error:   0
  1 =>   1
error:   0
  2 =>   2
error:   0
  3 =>   3
error:   0
  4 =>   4
error:   0
  5 =>   5
error:   0
```

```
140 => 140
error:   0
141 => 141
error:   0
142 => 142
error:   0
143 => 143
error:   0
144 => 144
error:   0
145 => 145
error:   0
146 => 146
error:   0
147 => 147
error:   0
148 => 148
error:   0
149 => 149
error:   0
Finish!!
  0 error detected!!
```

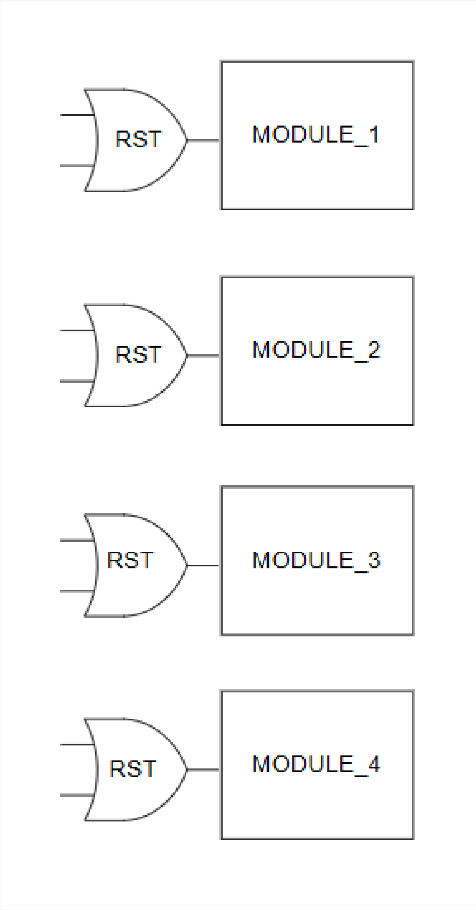
0~149까지 uart 입력

문제해결

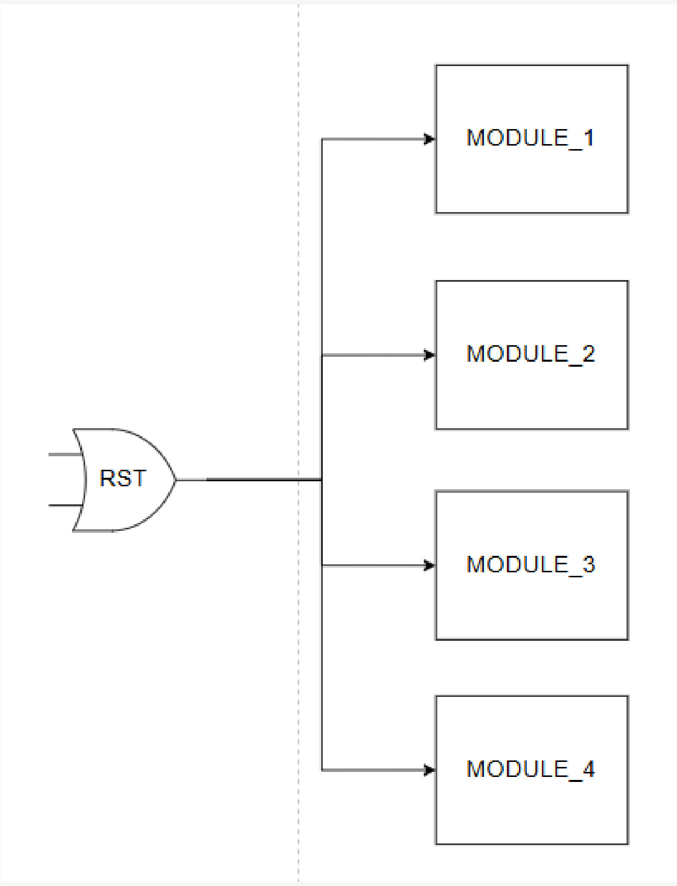
1. One-Wire 출력

→ Gate 수 감소

Problem



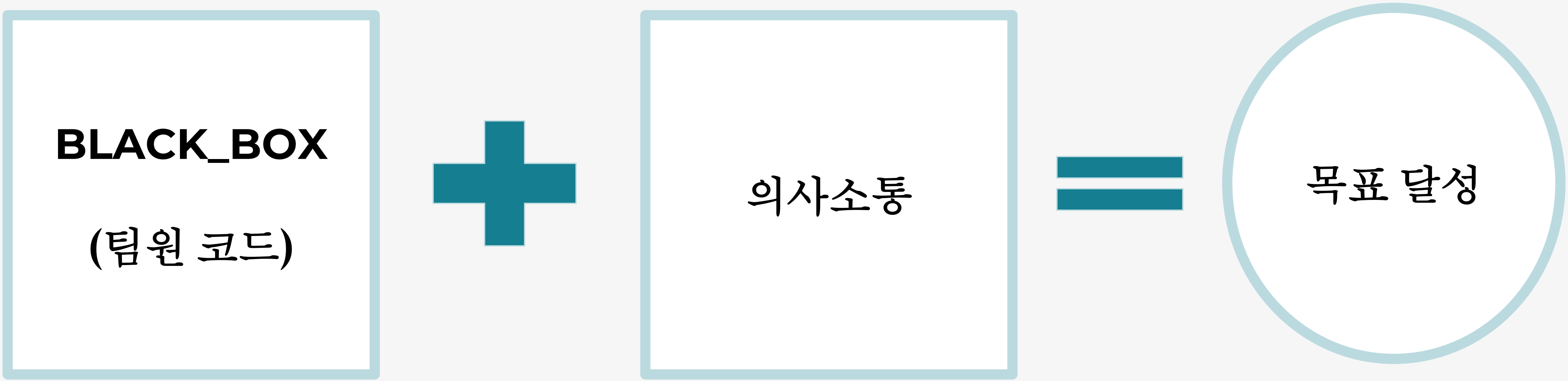
Solution



문제해결

2. 팀원과의 코드 공유

→ 꾸준한 의사소통으로 해결



BLACK_BOX

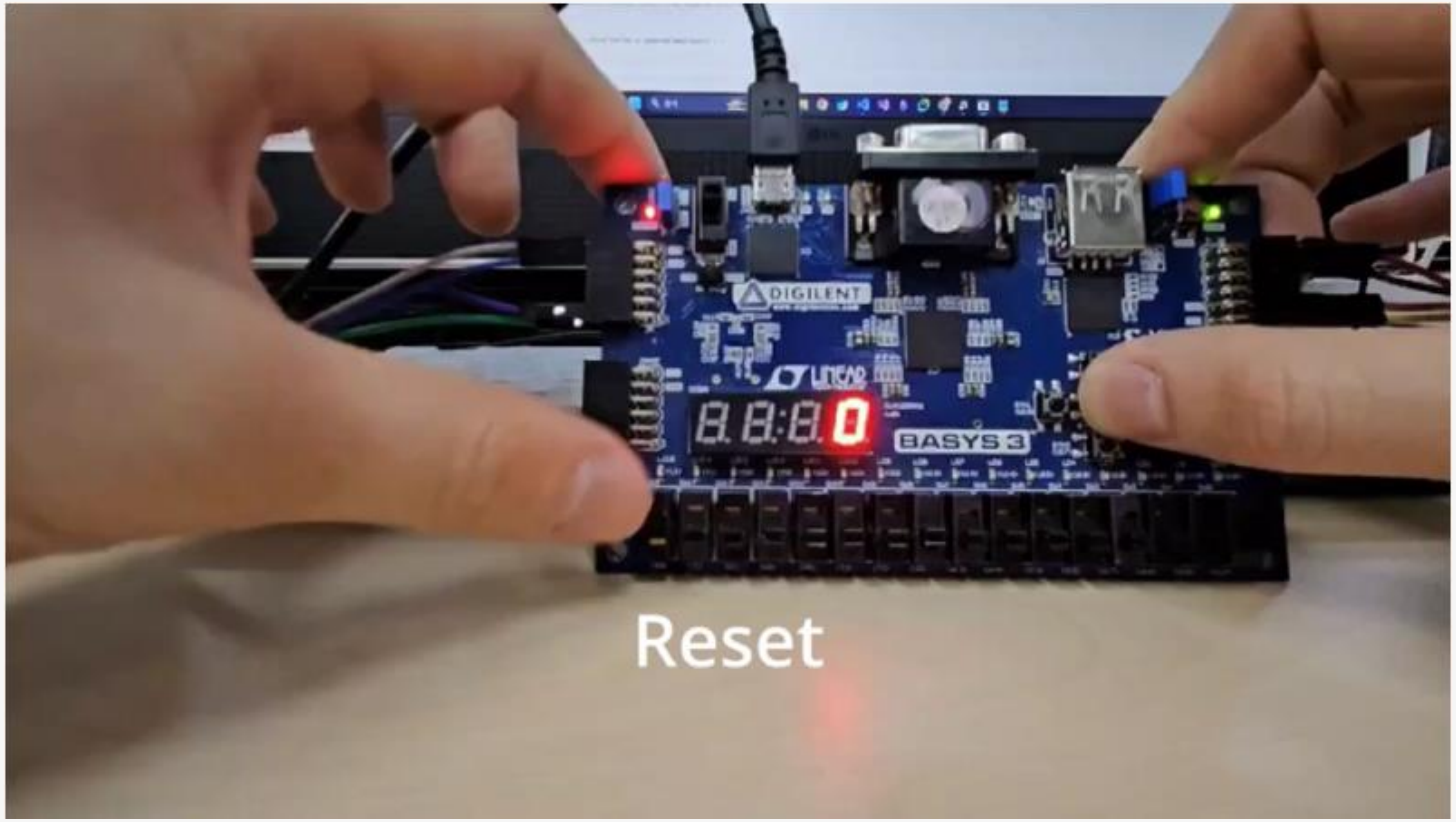
(팀원 코드)

의사소통

목표 달성

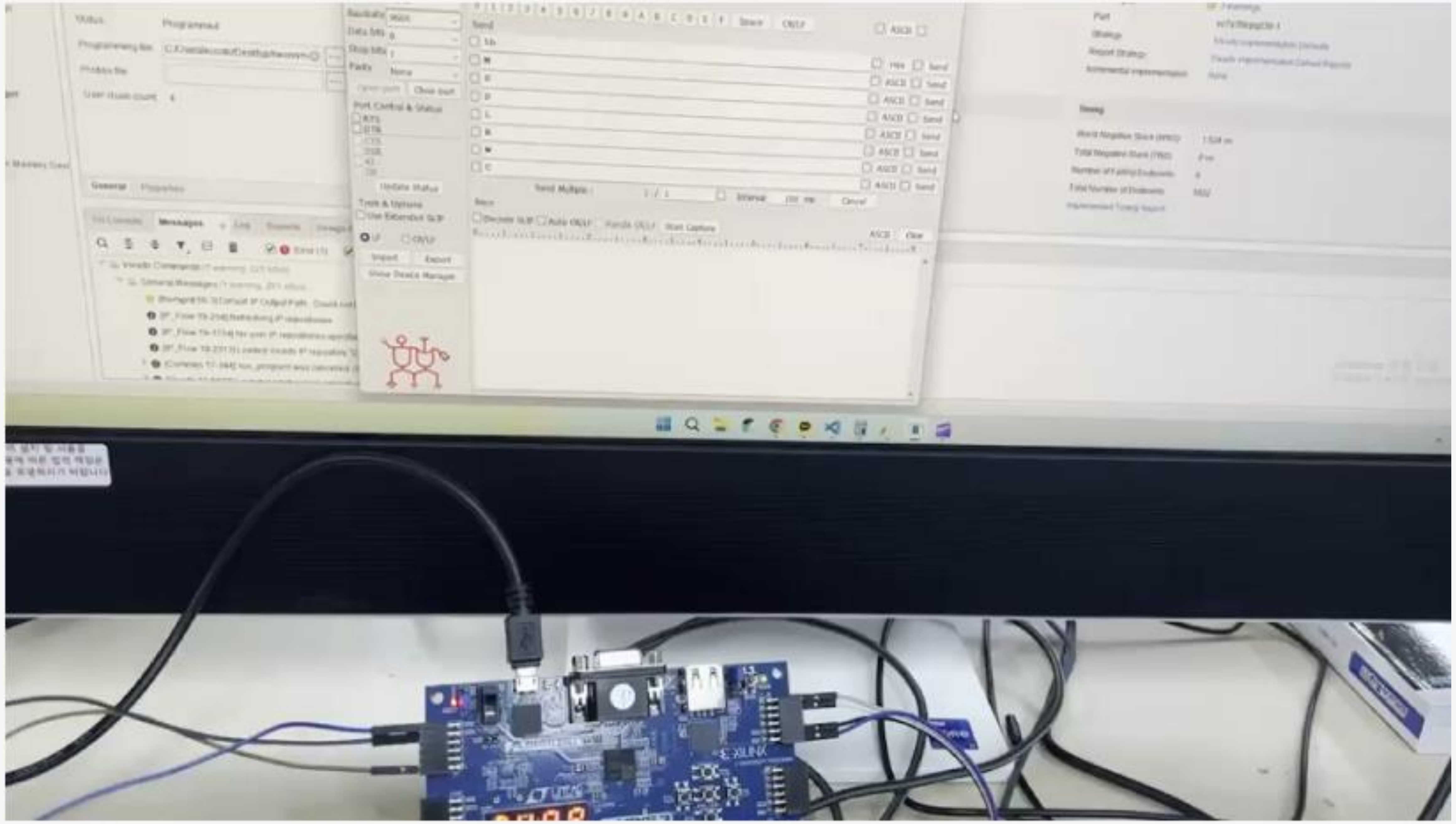
시연 영상

1. 버튼 제어



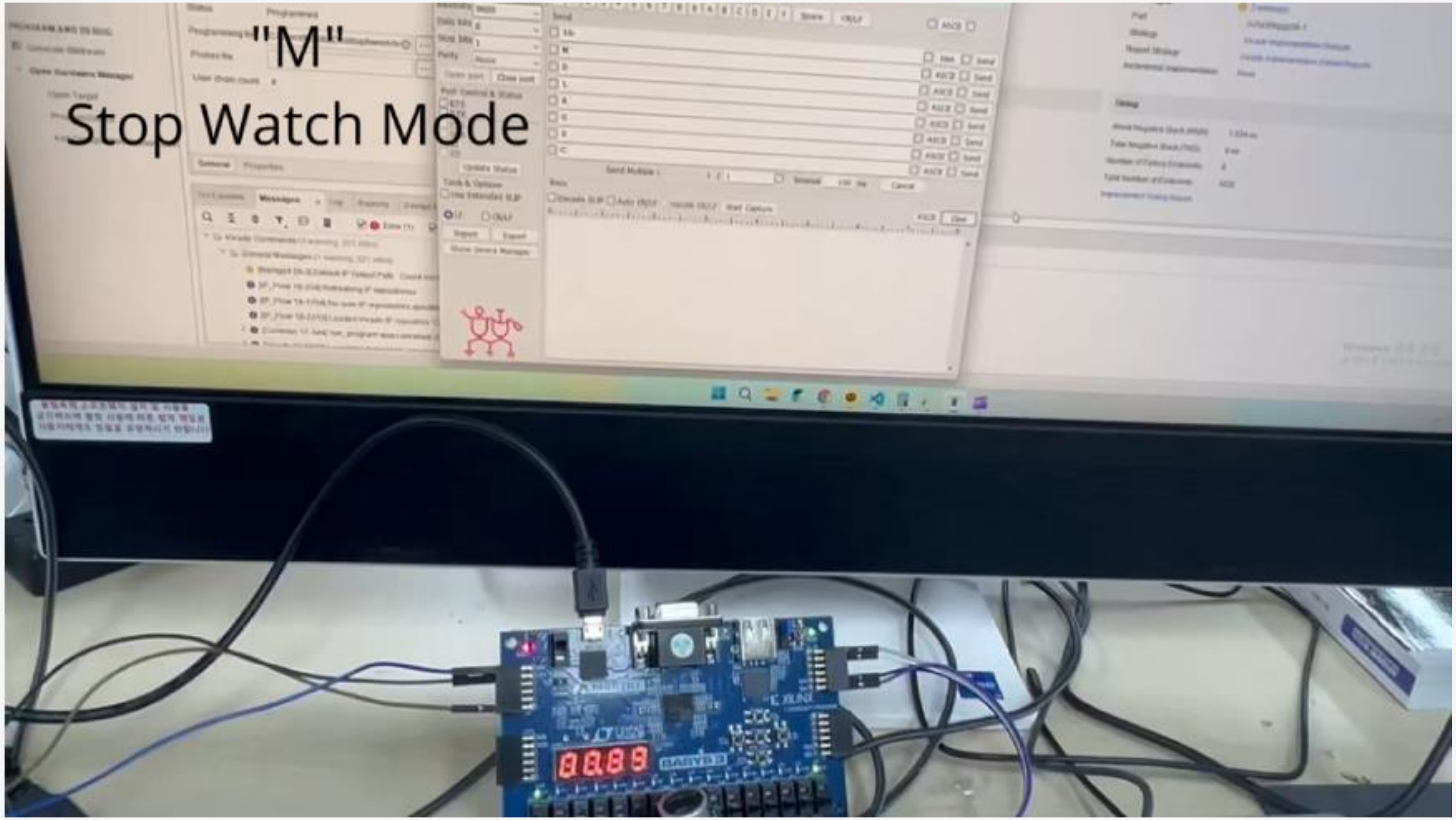
시연 영상

2. UART Watch



시연 영상

3. UART StopWatch



시연 영상

4. UART Sensor