온습도 센서 실습을 위한 *Verilog* 설계

(*DHT11*)

[*Reference*]

- *https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf*
- *https://blog.naver.com/PostView.nhn?blogId=compass1111&logNo=221213099193*

Temperature–Humidity Sensor Module (DHT11)

➢ Background

온도센서

- 서미스터(*Thermistor*)란 저항기의 일종으로, 온도에 따라 물질의 저항이 변화하는 성질을 이용한 전기적 장치이다.
- 열가변저항기라고도 하며, 주로 회로의 전류가 일정 이상으로 오르는 것을 방지하거나, 회로의 온도를 감지하는 센서로써 이용된다.
- 서미스터는 주로 폴리머나 세라믹 소재로 제작되며, 섭씨 영하 90도에서 130도 사이에서 높은 정확도로 온도를 측정할 수 있다. 이러한 점에서 순수한 금속을 사용하여 고온의 온도를 측정하는 저항 온도계와는 차이를 보인다.

[출처] 서미스터 - 위키백과, 우리 모두의 백과사전

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

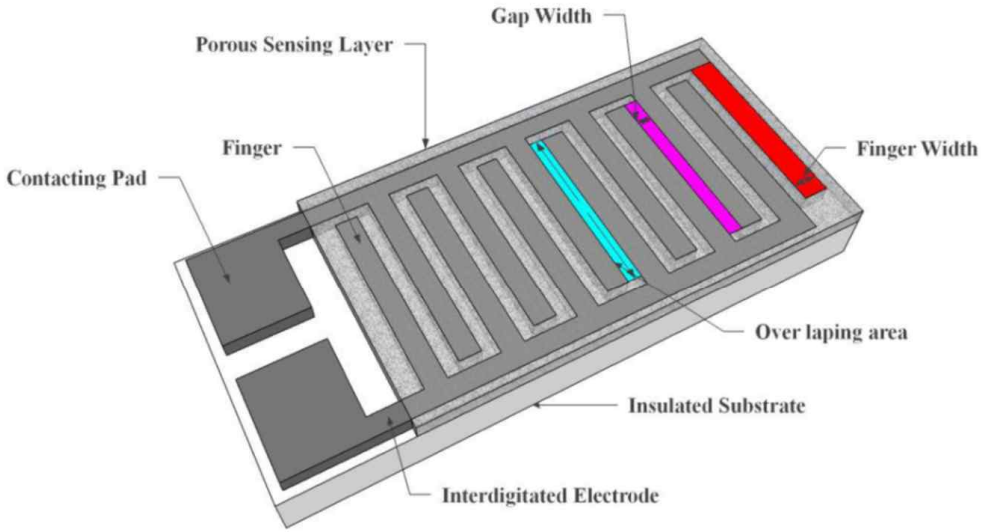
➢ Background

습도센서

❖ 습도센서는 아래와 같이 두 가지 종류

- 정전용량센서(***capacitive sensor***)는 두 개의 작은 막에 습도를 감지하여 정전용량을 체크해서 습도를 출력해주는 센서.
- 저항센서(resistive sensor)는 여러 빗살형 선들 사이에의 습도를 체크하여 저항값으로 습도를 출력해주는 센서.

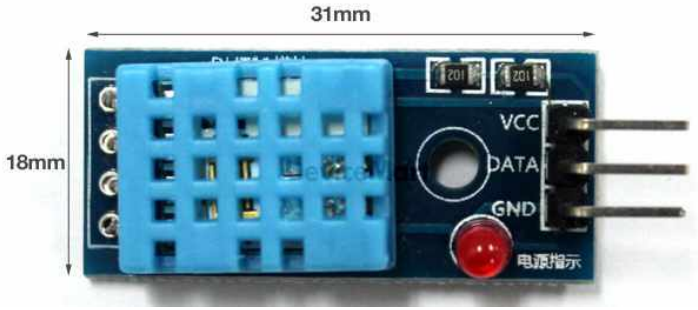
그림에서 ***Contacting pad***와 ***Interdigitated electrode***는 떨어져 있음. 센서 부분 여러 개의 빗살 패턴 사이에 습도가 들어가면 미세하게 전류가 흐를 수 있는 저항치가 발생하고 저항값이 증가/감소 하는 원리.



Temperature–Humidity Sensor Module(DHT11) ➢ *DHT11.xpr*

➢ 핀 구성 : ***DATA(DHT11), Vcc, Gnd***

- ***DATA*** : 온습도 센서 통신 신호 입출력 ***Pin*** (***Inout Port***)
- ***Vcc(3.3V)*** : ***3.3v Pmod*** 전원과 연결
- ***GND*** : ***BASYS3 Pmod***의 접지와 연결



Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Temperature–Humidity Sensor Module(DHT11) ➢ *DHT11.xpr*

➢ *Temperature–Humidity Sensor (DHT11)* 의 통신 원리(1)

- DHT11의 DATA 핀은 inout port에 할당되고, PULL UP을 위해 5k옴 저항을 이용.

  - 그림 2의 좌측 센서처럼 모듈화 되어있으면 모듈 내부에 풀업 저항이 존재하며, 우측 센서와 같으면 직접 풀업 저항을 연결해주어야 함.

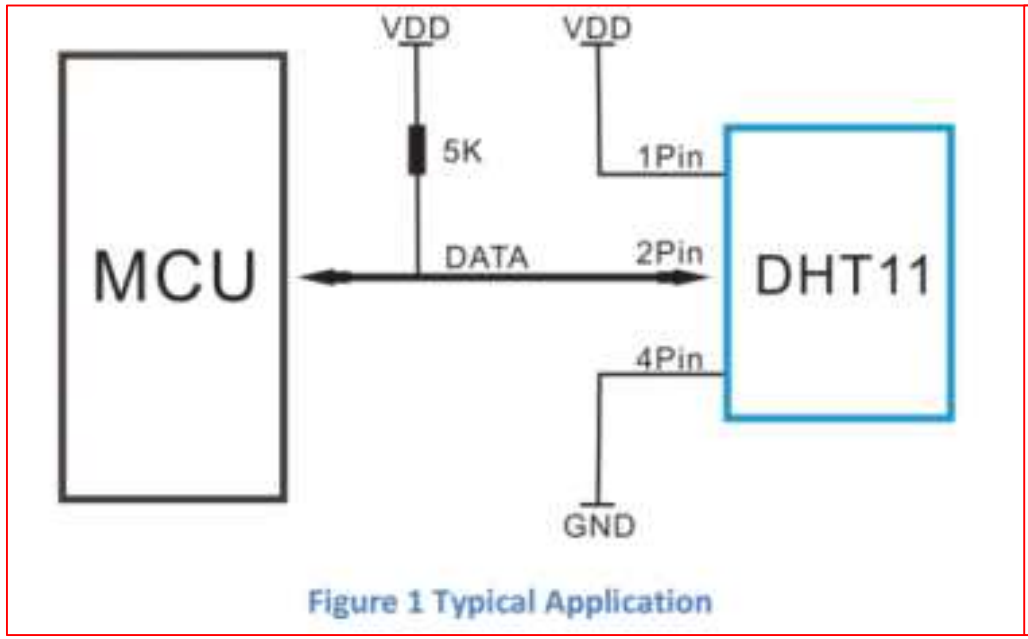
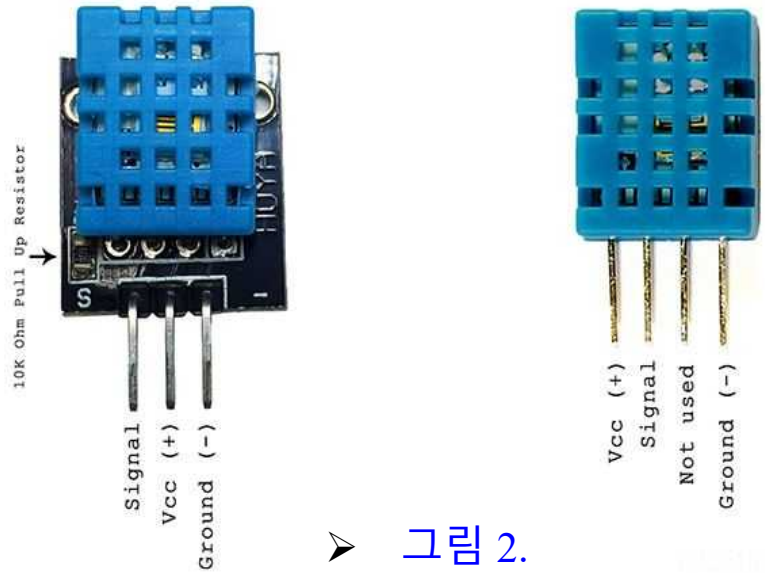


Figure 1 Typical Application



➢ 그림 2.

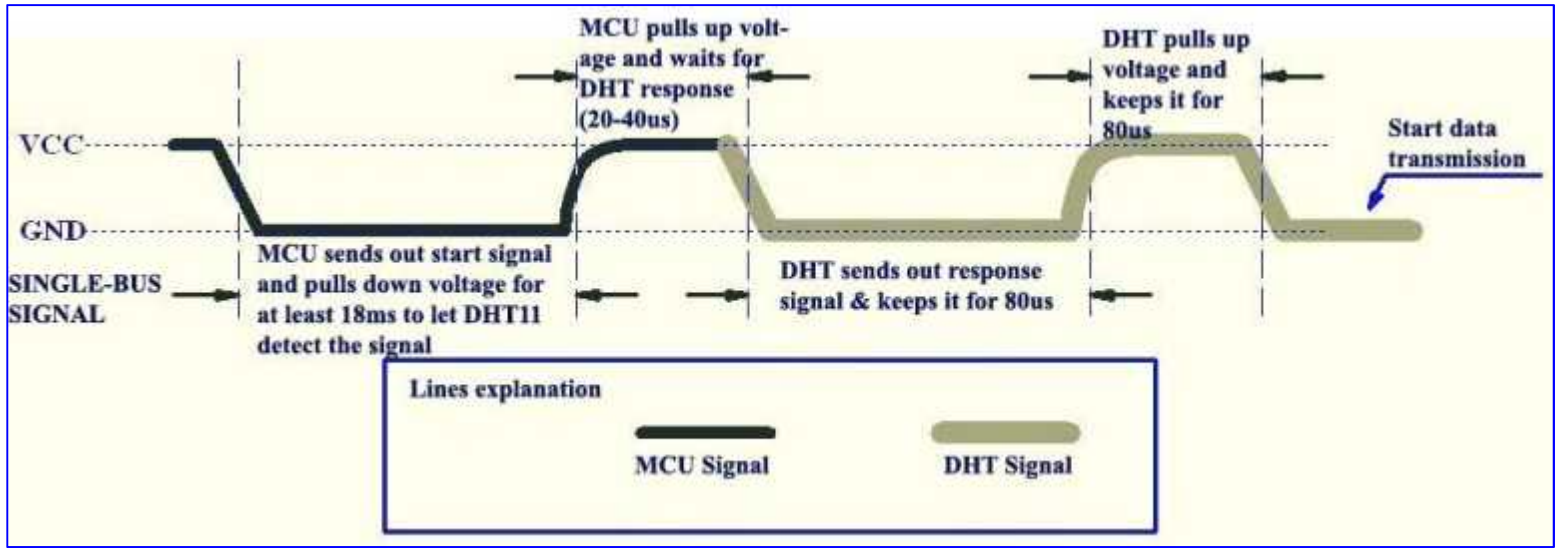
Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ Temperature–Humidity Sensor (DHT11) 의 통신 원리(2)

- DATA핀은 PULL UP 되어있으므로, 1의 Default 값을 가짐

- 제어부에서 DATA 핀에 LOW를 입력하여 DHT11 모듈에 Start Signal (Low, 18ms 이상)을 전송 후 HIGH 상태에서 대기 (20~40usec), 이 때 임피던스 출력(1'bz)을 하므로 HIGH까지 완만한 곡선이 그려진다.

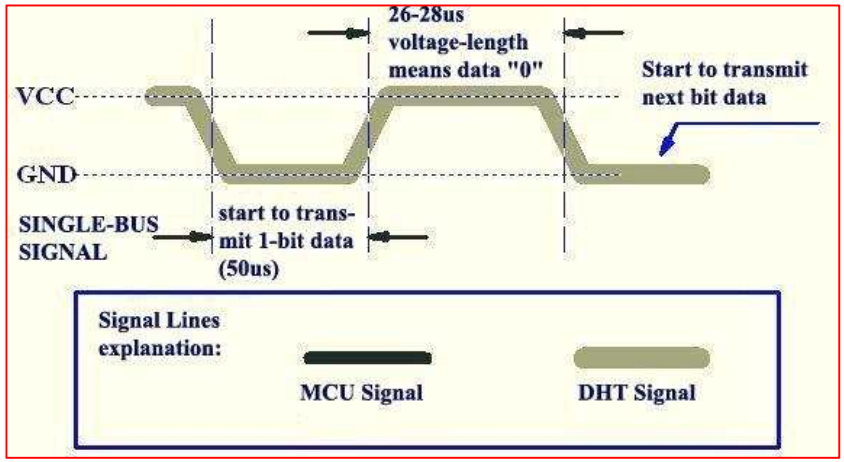
- 이후 DHT가 응답하면 80usec동안 LOW Signal, 80usec동안 HIGH Signal을 보내 Response Signal을 보낸 후 데이터 전송 시작.

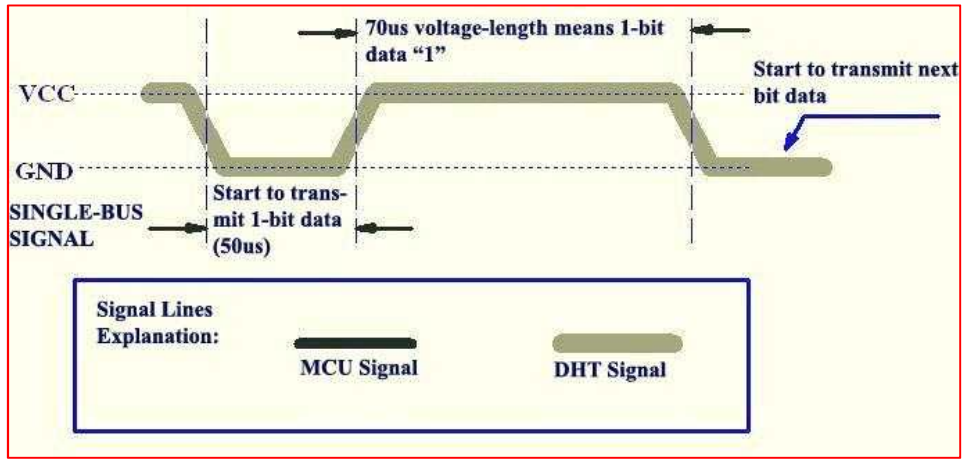


Temperature–Humidity Sensor Module(DHT11) ➢ *DHT11.xpr*

➢ *Temperature–Humidity Sensor (DHT11)* 의 통신 원리(3)

- 데이터 전송 시에 HIGH를 유지하는 시간을 통해 0과 1을 구분.
- 전송 시작 전 *LOW Signal 50usec* 유지
- 이후 약 50usec을 기준으로, ***HIGH***를 그보다 짧게 유지하면 0, 길게 유지하면 1을 출력한다.





- 데이터는 습도의 정수 부분 8Bits/소수 부분 8Bits, 온도의 정수 부분 8Bits/소수 부분 8Bits, 통신 확인 1Byte를 더해 총 ***40Bits***로 구성.

Data format: 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum. If the data transmission is right, the check-sum should be the last 8bit of "8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data".

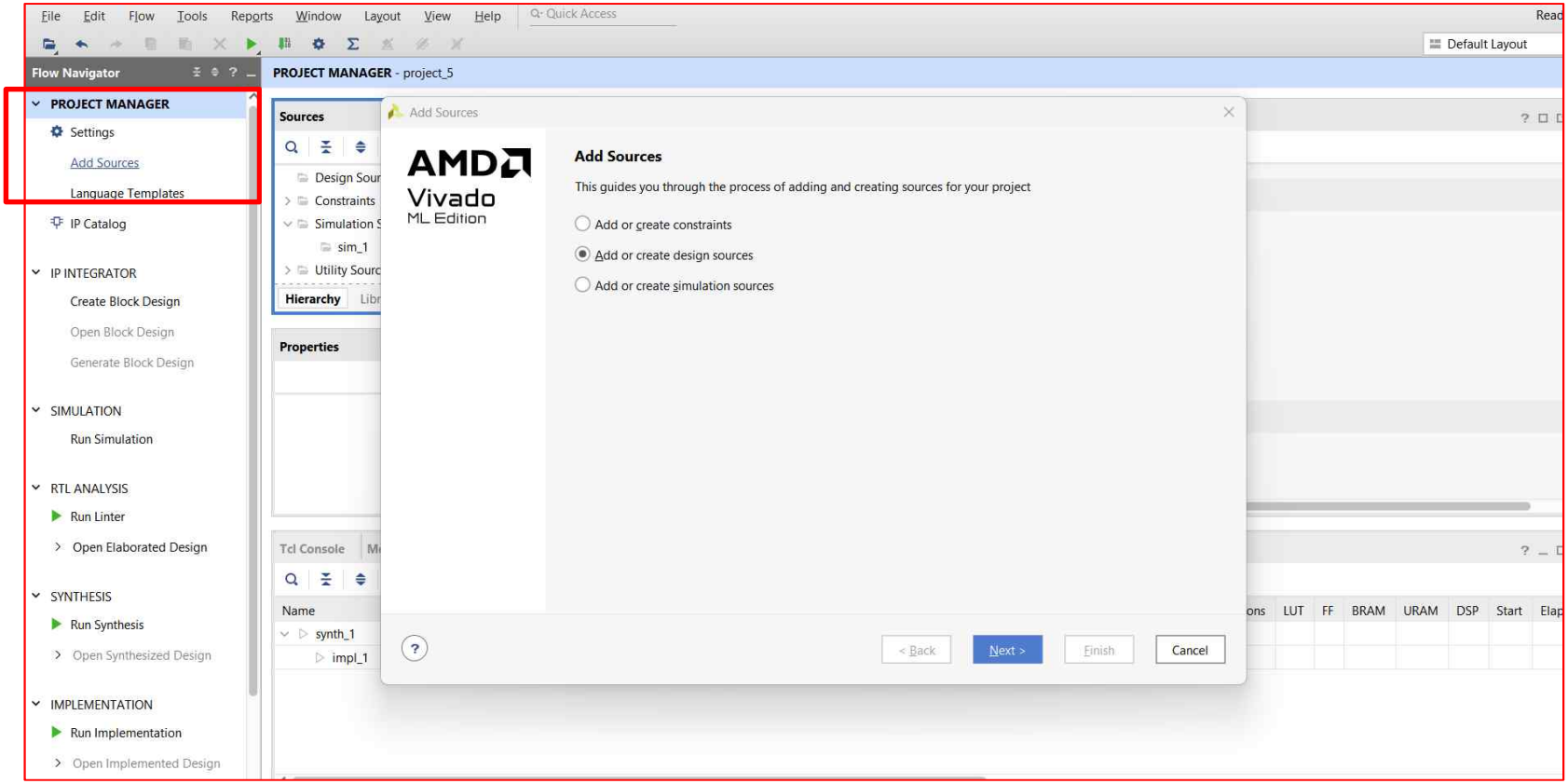
Temperature–Humidity Sensor Module (DHT11)

Verilog HDL Coding

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ MODULE 구현

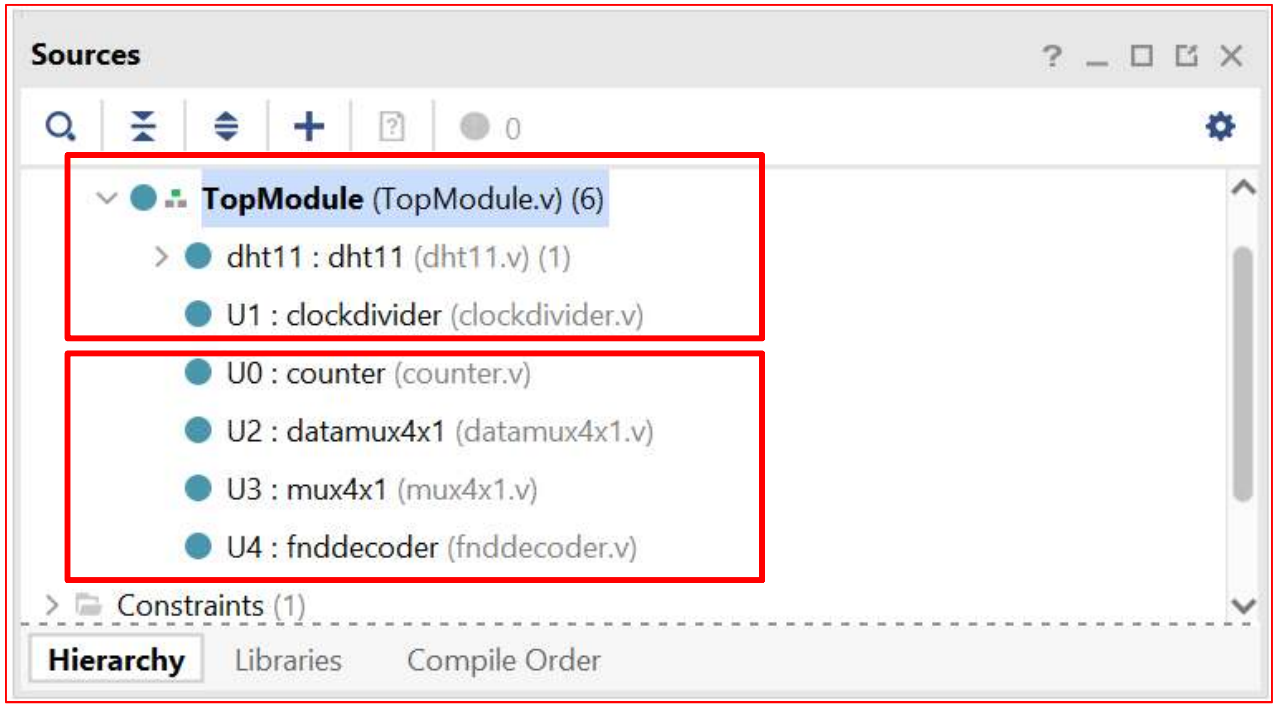
- *PROJECT MANAGER → Add Sources*



Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ MODULE 구현

- PROJECT MANAGER → Add Sources
- Create Source File → **TopModule.v, dht11.v, clockdivider.v, edge_detector.v 생성**
  → **fndcontrol 생성**



Temperature–Humidity Sensor Module(DHT11)

➢ *DHT11.xpr*

➢ ***MODULE*** 구현 → 온습도 센서 모듈 구동을 위한 ***Verilog HDL code***

dht11.v

```
1.  module DHT11(
2.  input clk, reset,clk1Mhz,
3.  inout dht11_data,
4.  output  [3:0] humidity10,humidity0,
5.  output  [3:0] temperature10,temperature0,
6.  output reg [7:0] led_bar
7.                  );

8.      reg [7:0] humidity,temperature;
9.
10.     assign humidity10 = humidity/10;
11.     assign humidity0 = humidity%10;
12.     assign temperature10 = temperature/10;
13.     assign temperature0 = temperature%10;

14.     parameter S_IDLE = 3'b000;
15.     parameter S_LOW_18MS = 3'b001;
16.     parameter S_HIGH_20US = 3'b010;
17.     parameter S_LOW_80US = 3'b011;

18.     parameter S_HIGH_80US = 3'b100;
19.     parameter S_READ_DATA = 3'b101;
20.     parameter S_WAIT_PEDG = 3'b110;
21.     parameter S_WAIT_NEDGE=3'b111;

22.      wire dht_nedge, dht_pedge;
23.      reg count_usec_e;
24.      reg [21:0] count_usec;
25.      reg [2:0] state, next_state, read_state;
26.      reg dht11_data_oe, dht_buffer;
27.      reg [5:0] data_count;
28.      reg [39:0] temp_data;

29.      assign dht11_data =  dht_buffer
```

❖ 1-29: 입출력 port 및 레지스터 선언

- ***clk***는 보드상에 장착된 100Mhz, reset은 negedge reset의 역할
- ***clk1Mhz*** : clockdivider에서 1Mhz clk(1usec) 생성
- ***Humidity10, Humidity0*** : 측정한 습도 출력
- ***Temperature10, temperature0*** : 측정한 온도 출력
- ***Led_bar*** : 온습도 센서 State

- Line 15 ~16 : DATA 핀에 LOW를 입력하여 DHT11 모듈에 Start Signal (Low, 18ms 이상)을 전송 후 HIGH 상태에서 대기 (20~40usec)
- Line 17 ~18 : 이후 DHT가 응답하면 80usec동안 LOW Signal, 80usec동안 HIGH Signal을 보내 Response Signal을 보낸 후 데이터 전송 시작
- HIGH를 유지하는 시간을 통해 0과 1을 구분.
- 전송 시작 전 LOW Signal 50usec 유지
- 이후 약 50usec을 기준으로, HIGH를 그보다 짧게 유지하면 0, 길게 유지하면 1을 출력

❖ 17-21: State 제어를 위한 parameter 선언

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ *MODULE* 구현 → 온습도 센서 모듈 구동을 위한 *Verilog HDL code*

dht11.v

```
30. always @(negedge clk1Mhz or negedge reset) begin
31.     if(~reset)  count_usec = 0;
32.     else begin

33.         if(count_usec_e)
34.             count_usec = count_usec + 1;

35.         else if(!count_usec_e) count_usec = 0;
36.         end

37.   end
```

❖ 30-37: 1Mhz 동기하여 시간 counting
- clk1Mhz는 Clockdivider 모듈에서 생성
- count_usec_e(count_usec의 Enable)가 0으로 초기화되면 카운팅 시간 초기화

```
38. always @(negedge clk1Mhz or negedge reset)begin
39.                 if(~reset)state = S_IDLE;
40.                 else state = next_state;
41.     end
42.
```

❖ 38-42: State 제어를 위한 반복구문 [1Mhz(1usec) 동기]

```
43.     edge_detector ed_dec
    (.clk(clk1Mhz), .cp_in(dht11_data), .reset(reset), .n_
    edge(dht_nedge), .p_edge(dht_pedge));
```

❖ 43: *DHT11 data edge* 검출을 위한 *edge_detector*

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ ***MODULE*** 구현 → 온습도 센서 모듈 구동을 위한 ***Verilog HDL code***

dht11.v

```
44.  always @(posedge clk1Mhz or negedge
     reset)begin

45.     if(~reset)begin
46.             count_usec_e <= 0;
47.             next_state <= S_IDLE;
48.             dht11_data_oe <= 0;
49.             dht_buffer <= 1'bz;
50.             read_state <= S_WAIT_PEDGE;
51.             data_count <= 0;
52.             led_bar = 8'b1111_1111;
53.             end
54.     else begin

55.     case(state)
56.     S_IDLE : begin
57.     led_bar[0] <= 0;
58.     if(count_usec< 22'd3_000_000) begin

59.             count_usec_e <= 1;
60.             dht_buffer = 1'bz;
61.             end

62.             else begin
63.             led_bar <= 8'b1111_1111;
64.             next_state <= S_LOW_18MS;
65.             count_usec_e <= 0;
66.             end
67.         end
68.
```

❖ 44-170: DHT11 DATA 검출을 위한 State 제어문
- Clk 1Mhz에 동기하여 동작

❖ 45-53: 리셋 동작시 레지스터 모두 초기화

[Example]

assign out = (cs) ? in : 1'bz;

➔ *cs가 0 → 1로 바뀌면 out의 값이 in 값으로 동기화되지만, 1 → 0으로 바뀌면 고임피던스를 가지지 않고 이전 값을 가짐*

[0] out = x cs = 0 in = 0

[1] out = 0 cs = 1 in = 0

[2] out = 0 cs = 0 in = 0

[3] out = 0 cs = 0 in = 1

[4] out = 1 cs = 1 in = 1

[5] out = 1 cs = 0 in = 1

❖ **56-68: IDLE**
- LED 0번 OFF
- 58-61: 3sec동안 IDLE 상태에 대기, DHT는 임피던스 출력
- 64-67: 3sec지나면 LED 초기화 후 새로운 DHT data 받아옴 (Data가 3초마다 갱신)

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ *MODULE* 구현 → 온습도 센서 모듈 구동을 위한 *Verilog HDL code*

dht11.v

```
69.  S_LOW_18MS : begin
70.       led_bar[1] <= 0;
71.      if(count_usec < 22'd20_000) begin
72.                  count_usec_e <= 1;
73.                  dht_buffer <= 0;
74.                  end
75.      else begin
76.                  count_usec_e <= 0;
77.                  next_state <= S_HIGH_20US;
78.                   dht_buffer = 1'bz;
79.                  end
80.      end
```

❖ **69-80:** ***LOW_18ms (start signal)***

- LED 1번 OFF
- 71-74: DHT data에 20ms 동안 LOW 출력 (Start Signal)
  (18ms에 여유를 둠)
- 79-79: DHT에 임피던스 출력하면서 다음 State

```
81.  S_HIGH_20US : begin
82.                  led_bar[2] <= 0;
83.      if(count_usec < 22'd70) begin
84.                  dht_buffer = 1'bz;
85.                  count_usec_e <= 1;
86.          if(dht_nedge)begin
87.                  next_state <= S_LOW_80US;
88.                  count_usec_e <= 0;
89.          end
90.  else begin next_state <= S_HIGH_20US;
91.                  count_usec_e <= 1;
92.                  end
93.      end
94.   else begin
95.      next_state <= S_IDLE;
96.      count_usec_e <= 0;
97.      end
98.   end
```

❖ **81-98:** ***HIGH_20us*** **(DHT 응답 대기)**

- LED 2번 OFF
- 70usec동안 임피던스 출력하며 DHT의 응답 대기.
  (20usec에 여유를 둠)
- 86-92: DHT가 LOW로 응답하면 다음 State, 70usec 동안 동작
- 94-97: 응답이 없으면 IDLE 상태 복귀

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ ***MODULE*** 구현 → 온습도 센서 모듈 구동을 위한 ***Verilog HDL code***

dht11.v

```
99.  S_LOW_80US : begin
100.    led_bar[3] <= 0;
101.    if(count_usec < 22'd90)begin

102.       if(dht_pedge)begin
103.          next_state <= S_HIGH_80US;
104.          count_usec_e <= 0;
105.          end

106.    else begin
107.    next_state <= S_LOW_80US;
108.    count_usec_e <= 1;
109.         end
```

```
110.    end   else begin
111.    next_state <= S_IDLE;
112.    count_usec_e <= 0;
113.    end
114.    end
```

❖ **99-109: *LOW_80us (DHT11 Response Signal)***
- LED 3번 OFF
- 102-109: DHT11의 응답을 검출하면 다음 State (80usec에 여유를 둠)

❖ 110-114: 통신 시간 내에 응답하지 못하면 IDLE로 초기화

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ *MODULE* 구현 → 온습도 센서 모듈 구동을 위한 *Verilog HDL code*

dht11.v

❖ **115-126:** ***HIGH_80us***
- LED 4번 OFF
- 118-126: DHT data가 LOW를 출력하면 (DHT 모듈이 데이터 전송 시작) DATA READ 시작

```
115.  S_HIGH_80US : begin
116.    led_bar[4] <= 0;
117.    if(count_usec < 22'd90)begin   // 80us동안 dht가 50us동안 start to transmit signal Low Voltage를 출력 할 때 negedge가 발생

118.      if(dht_nedge)begin
119.    next_state <= S_READ_DATA;      // 데이터 READ 상태 전환
120.    count_usec_e <= 0;              // 카운트 중단
121.    end

122.    else begin
123.    next_state <= S_HIGH_80US;      // 응답이 올 때 까지 현재 상태 유지
124.    count_usec_e <= 1;              // 카운트 지속
125.    end
126.    end
```

```
127.    else begin
128.    next_state <= S_IDLE;           // 80us후 IDLE 상태 전환, 카운트 중단
129.    count_usec_e <= 0;
130.    end
131.    end
```

❖ 127-131: 통신 시간 내에 응답하지 못하면 IDLE로 초기화

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ *MODULE* 구현 → 온습도 센서 모듈 구동을 위한 *Verilog HDL code*

dht11.v

```
132. S_READ_DATA : begin      // start_to_transmit 1-bit data
133.                led_bar[5] <= 0;
134.    case(read_state)
135.      S_WAIT_PEDGE: begin
136.           if(dht_pedge) begin
137.                read_state <= S_WAIT_NEDGE; // negative edge 대기 상태로 전환
138.                count_usec_e <= 1;          // 카운트 시작
139.           end
140.           else begin
141.               count_usec_e = 0;
142.           end
143.     end

144.   S_WAIT_NEDGE: begin
145.       if(dht_nedge) begin
146.           data_count <= data_count + 1;      // 40bit 까지 카운트 하기 위해 data가 들어온 횟수를 카운트
147.           read_state <= S_WAIT_PEDGE;       // positive edge로 voltage_length 대기

148.           if(count_usec < 50) begin
149.           temp_data <= {temp_data[38:0],  1'b0};          // 좌 시프트, 50us 이하면 0 저장
150.           end
151.           else begin
152.           temp_data <= {temp_data[38:0], 1'b1};           // 50us 이상이면 1 저장
153.           end
154.   end
155.   else begin
156.   count_usec_e <= 1;
157.   read_state <= S_WAIT_NEDGE;
158.   end
159.   end
160.   default: read_state = S_WAIT_PEDGE;
161.   endcase
```

❖ **132-177: *READ DATA***

- LED 5번 OFF
- 148-153: High 신호가 50usec보다 길면 1, 짧으면 0으로 계산하여 Data를 Read
- temp_data에 Read한 Data 저장

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ ***MODULE*** 구현 → 온습도 센서 모듈 구동을 위한 ***Verilog HDL code***

dht11.v

- ❖ 162-165: 데이터 통신이 완료되면 IDLE 상태로 복귀
- ❖ 166-168: 검증용 데이터가 측정값과 일치하면 온/습도 데이터 출력

```
162.if(data_count >= 40)begin
163.    led_bar[6] <= 0;
164.    data_count <= 0;
165.    next_state <= S_IDLE;

166. if(temp_data[39:32] + temp_data[31:24] + temp_data[23:16] + temp_data[15:8] == temp_data[7:0]) begin

167.   humidity <= temp_data[39:32];
168.   temperature <= temp_data[23:16];
169.      end
170.    end
171.   end
172.    default : next_state = S_IDLE;
173.   endcase
174.    end
175.    end
176.
177.endmodule
```

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

➢ *MODULE* 구현 → 온습도 센서 모듈 구동을 위한 *Verilog HDL code*

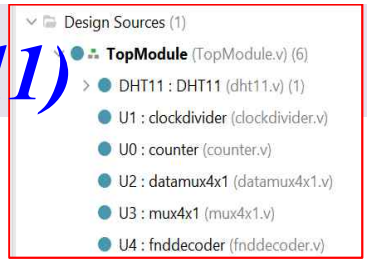
edge_detector.v

```
1.  `timescale 1ns / 1ps
2.  module edge_detector(
3.  input clk,
4.  input cp_in,
5.  input reset,
6.  output p_edge,
7.  output n_edge
8.      );
9.
10.     reg cp_in_old, cp_in_cur;
11.
12.     always@(posedge clk or negedge reset) begin
13.     if(~reset) begin
14.     cp_in_old=0; cp_in_cur = 0;
15.     end else begin
16.
17.     cp_in_old <= cp_in_cur;
18.     cp_in_cur <= cp_in;
19.     end
20.  end
21.
22.     assign p_edge = ~cp_in_old & cp_in_cur;
23.     assign n_edge = cp_in_old & ~cp_in_cur;
24.
25. endmodule
```

❖ 1-9: 입출력 port 및 레지스터 선언
- clk는 보드상에 장착된 100Mhz, reset은 negedge reset의 역할
- cp_in : 엣지를 검출하고자 하는 신호. 해당 모듈에서는 echo

❖ 12-25
- cp_in_old 에는 이전의 cp_in값 저장
- cp_in_cur 에는 현재의 실시간 cp_in값 저장
- p_edge : cp_in이 라이징 엣지일 때 1
- n_edge : cp_in이 다운 엣지일 때 1

Temperature–Humidity Sensor Module(DHT11)



➢ *TopModule, clockdivider*

TopModule.v (1)

```
1.  `timescale 1ns / 1ps
2.  module TopModule(
3.  input clk,
4.  input reset,
5.  inout dht11_data,
6.  output [7:0] fnd,
7.  output [3:0] fndsel,
8.  output [7:0] led_bar
9.  );

10. DHT11 DHT11(
11. .clk(clk),
12. .reset(reset),
13. .clk1Mhz(clk1Mhz),
14. .dht11_data(dht11_data),
15. .humidity10(ind),
16. .humidity0(inc),
17. .temperature10(inb),
18. .temperature0(ina),
19. .led_bar(led_bar)
20.  );
21.

22. wire [3:0] ina,inb, inc, ind;

23. clockdivider U1(
24. .clk(clk),
25. .reset(reset),
26. .clk1Mhz(clk1Mhz),
27. .clk1000hz(w_clkout)
28. );
```

TopModule.v (2)

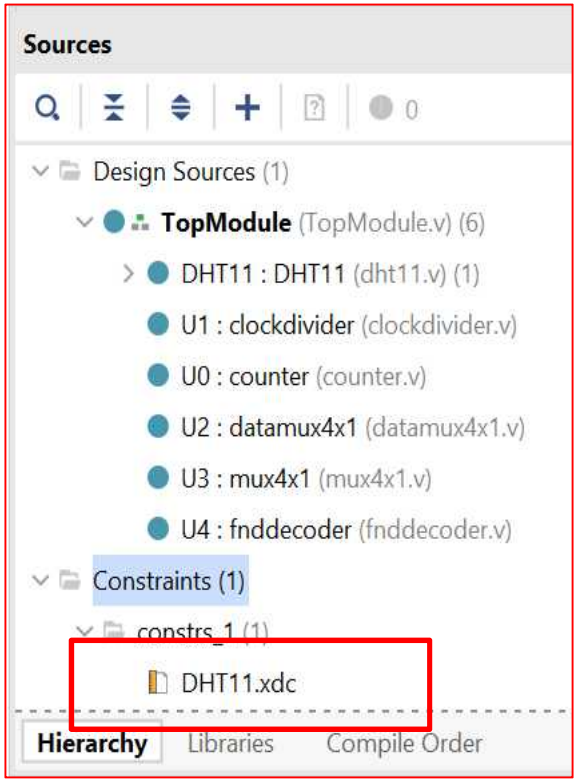
```
29. wire  w_clkout;
30. wire [1:0] w_counter;
31. wire [3:0] w_datafnd;

32.    counter U0 (
33.     .inclk(w_clkout),
34.     .reset(reset),
35.     .out_counter(w_counter)
36.  );

37.    datamux4x1 U2 (
38.     .ina  (ina),
39.     .inb  (inb),
40.     .inc  (inc),
41.     .ind  (ind),
42.     .insel(w_counter),
43.     .outy  (w_datafnd)
44.  );
45.   mux4x1 U3 (
46. .s(w_counter),
47.     .y(fndsel)
48.  );
49.
50.   fnddecoder U4 (
51. .a(w_datafnd),
52.     .fnd(fnd)
53.  );


54. endmodule
```

clockdivider.v

```
1.  `timescale 1ns / 1ps
2.  module clockdivider (
3.    input  clk,
4.    input  reset,
5.    output reg  clk1Mhz,
6.    output reg  clk1000hz
7.  );
8.   reg [25:0] cnt = 0;
9.   always @(posedge clk or negedge reset) begin
10.    if (~reset) begin
11.      cnt <= 0;
12.      clk1Mhz <= 0;
13.    end else begin
14.      if (cnt == (50 - 1)) begin
15.        cnt <= 0;
16.        clk1Mhz <= ~clk1Mhz;
17.      end else begin
18.        cnt <=  cnt + 1;
19.      end
20.    end
21.  end
22.
23.   reg [25:0] cnt1 = 0;
24.  always @(posedge clk or negedge reset) begin
25.    if (~reset) begin
26.      cnt1 <= 0;
27.      clk1000hz <= 0;
28.    end else begin
29.      if (cnt1 == (50000 - 1)) begin
30.        cnt1 <= 0;
31.        clk1000hz <= ~clk1000hz;
32.      end else begin
33.        cnt1 <=  cnt1 + 1;
34.      end
35.    end
36.  end
37. endmodule
```

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

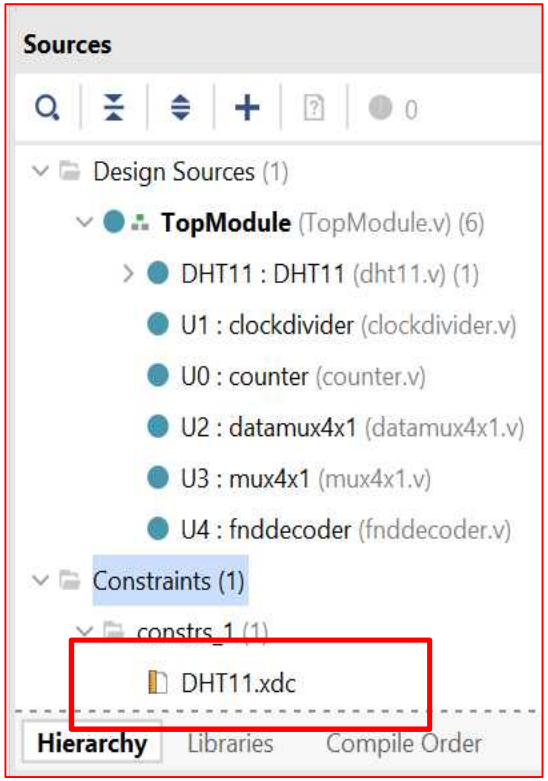
✓ *FND 모듈 추가 및 xdc pin-mapping 완료 후 실습*




```
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property -dict { PACKAGE_PIN R2    IOSTANDARD LVCMOS33 } [get_ports {reset}]

##7 Segment Display
set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {fnd[0]}]
set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {fnd[1]}]
set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {fnd[2]}]
set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {fnd[3]}]
set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {fnd[4]}]
set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {fnd[5]}]
set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {fnd[6]}]
set_property -dict { PACKAGE_PIN V7   IOSTANDARD LVCMOS33 } [get_ports {fnd[7]}]

set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {fndsel[0]}]
set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {fndsel[1]}]
set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {fndsel[2]}]
set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {fndsel[3]}]

## LEDs
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {led_bar[0]}]
set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports {led_bar[1]}]
set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports {led_bar[2]}]
set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports {led_bar[3]}]
set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports {led_bar[4]}]
set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports {led_bar[5]}]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {led_bar[6]}]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {led_bar[7]}]
```

Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

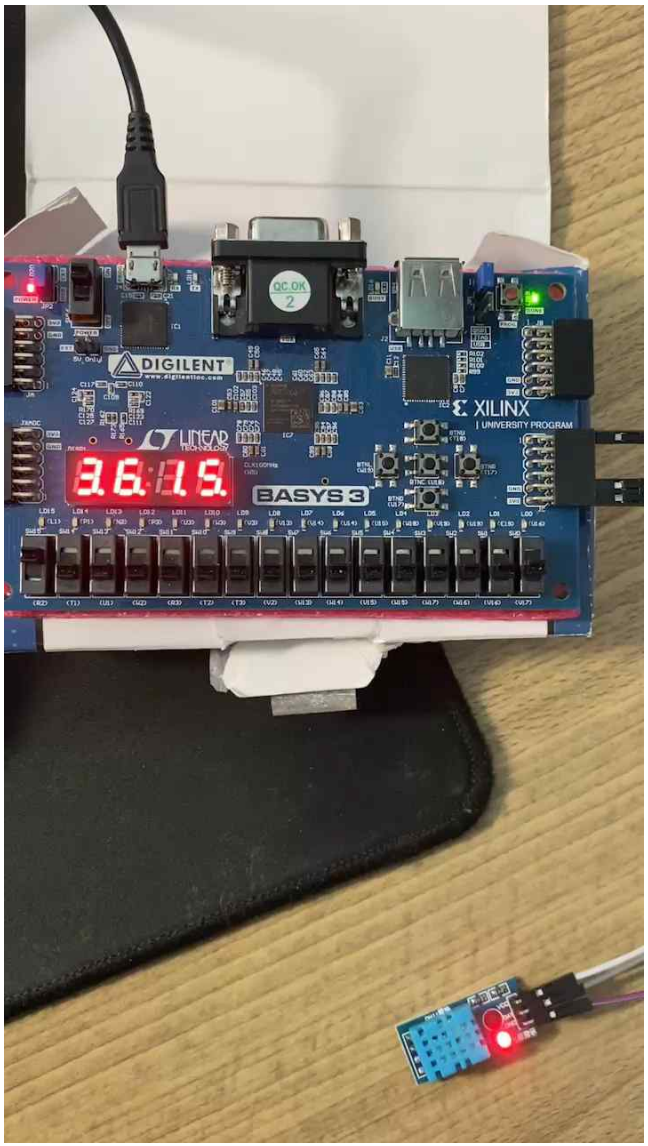
✓ *FND* 모듈 추가 및 *xdc pin-mapping* 완료 후 실습

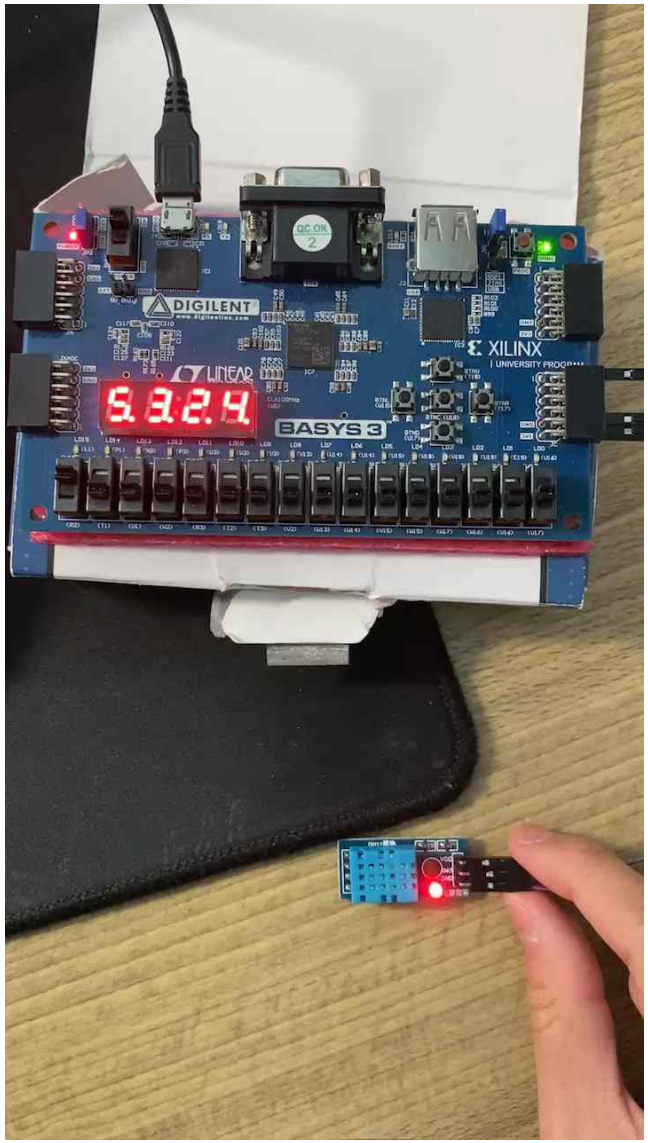



```
##Pmod Header JC
set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 } [get_ports
{dht11_data}];#Sch name = JC1


##USB-RS232 Interface
set_property -dict { PACKAGE_PIN B18   IOSTANDARD LVCMOS33 } [get_ports usb_uart_rxd]
set_property -dict { PACKAGE_PIN A18   IOSTANDARD LVCMOS33 } [get_ports usb_uart_txd]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]


## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

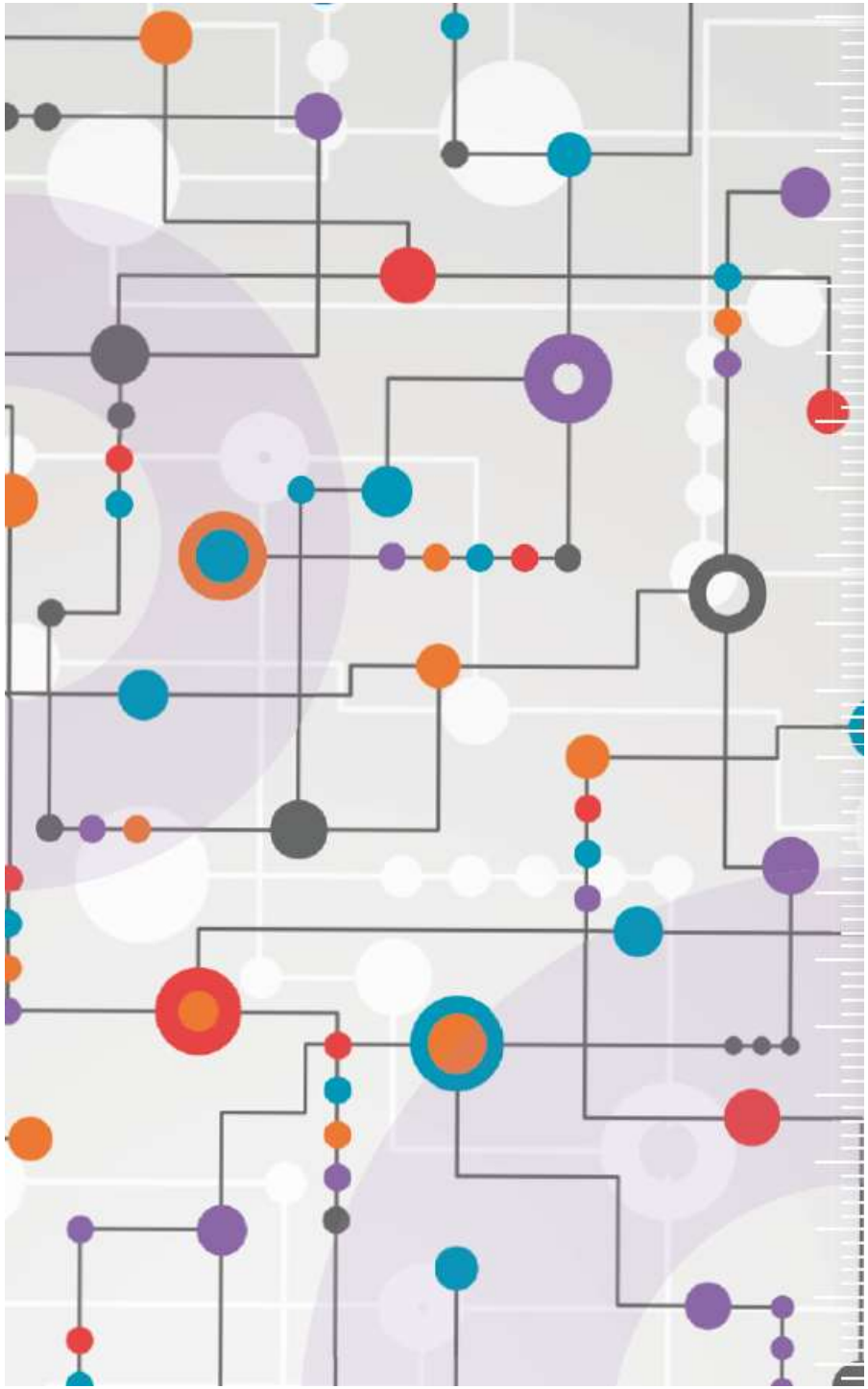
Temperature–Humidity Sensor Module(DHT11) ➢ DHT11.xpr

✓ *FND* 모듈 추가 및 *xdc pin-mapping* 완료 후 실습





✓ 센서에 입김을 불어넣었을 때, 센서 값이 변화하는 것을 확인할 수 있다.



수고하셨습니다.