

**SKEL4273 CAD WITH HDL**

**MILESTONE 5 & 6: CPLD PROJECT**

**DIGITAL THERMOMETER**

**LECTURER: MUHAMMAD MUN’IM BIN AHMAD ZABIDI**

GROUP NOVICHOK:

1. ONG CHEE SENG (A18KE0246)
2. MOH WEI JUN (A18KE0113)
3. **INTRODUCTION**

Nowadays, in the era of digitalization, digital thermometer has replaced traditional temperature sensing thermometer. This is because traditional temperature sensing thermometers are not able to fulfil the high demand and requirement of efficiency and accuracy temperature measurement.

Digital temperature measurement is very important physical parameter in the industry as well as in modern advance technology research and development (R&D). Digital thermometer is an instrument used to measure the temperature. According to the Science Direct, a digital thermometer is used to verify a smart temperature transmitter under flowing conditions and a successful calibration of the smart temperature transmitter.

The DHT11 is a commonly used as the temperature and humidity sensor in a simple circuit because the sensor can measure the temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. In term of power consumption, DHT11 has low power consumption and it is ultra-small size. The DHT11 sensor comes with a dedicated NTC to measure the surrounding temperature and humidity as well as an 8-bit microcontroller to output the values of temperature and humidity.

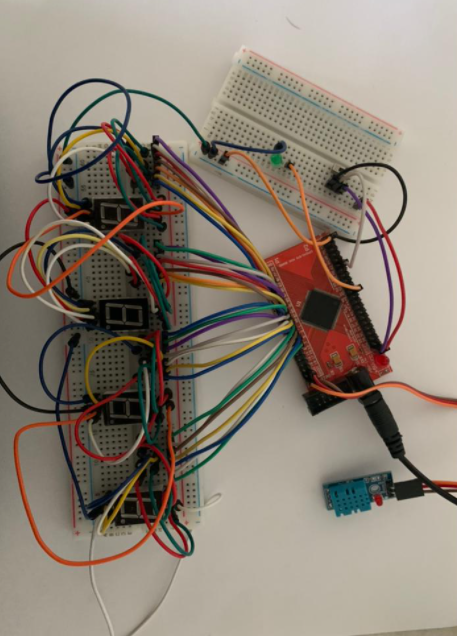
1. **PROJECT DESCRIPTION**

In this digital thermometer project, DHT11 is used to act as our temperature and humidity sensor. The purpose of this project is to measure the temperature and humidity in our house. When the surrounding temperature exceed 30°C, a LED will light up to give us a signal to switch on the air-conditioner. Surrounding humidity is not a consideration in giving signal to switch on the are conditioner. So, this is the objective of this project.

Quartus II software is used to implement this digital thermometer project on CPLD. Verilog code is used to design the all the Datapath Unit, Control Unit, Top Level Module as well as testbench (to verify the DHT11 Verilog Code). All the Verilog code can be referred in APPENDIX. For DHT11, There are 7 states in DHT11 sensor, which is

1. S\_POWER\_ON
2. S\_LOW\_20MS
3. S\_HIGH\_20US
4. S\_LOW\_83US
5. S\_HIGH\_87US
6. S\_SEND\_DATA
7. S\_DELAY
8. **SOFTWARE**
9. Quartus II
10. **HARDWARE**
11. DHT11 sensor
12. CPLD
13. Green LED
14. Four 7-segment LED display
15. Breadboard
16. Male-to-male jumpers
17. Male-to-female jumpers

Next, in order to show the actual value of surrounding temperature and humidity, four 7 segment LED display is used in this project. The circuit design is shown in Figure 1.



7-segment LED display is common-cathode type

Green LED indicated TH output.

Figure 1: The circuit design of Digital Thermometer.

The left two most of 7-segment LED display is used to display the value of surrounding humidity, while the right two most of 7-segment LED display is used to display the value of surrounding temperature. All the four 7-segment LED display used in this project is common-cathode type.

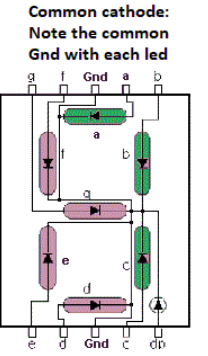


Figure 2: Common cathode 7-segment LED display.

1. **SIMULATION AND RESULTS**

**Part I: Simulation by using Quartus II Software**

Verilog Code and Testbench of Digital Thermometer Circuit

1. Datapath Unit

module DU1(

input clk,rst,us\_clear，

output reg[4:0] clk\_cnt,

output reg[20:0] count\_1us,

output reg clk\_1M

);

//to produce clock with 1us

always @ (posedge clk or negedge rst) begin

if (!rst) begin

clk\_cnt <= 5'd0;

clk\_1M <= 1'b0;

end

else if (clk\_cnt < 5'd24)

clk\_cnt <= clk\_cnt + 1'b1;

else begin

clk\_cnt <= 5'd0;

clk\_1M <= ~ clk\_1M;

end

end

//1us counter

always @ (posedge clk\_1M or negedge rst) begin

if (!rst)

count\_1us <= 21'd0;

else if (us\_clear)

count\_1us <= 21'd0;

else

count\_1us <= count\_1us + 1'b1;

end

endmodule

1. Control Unit

module CU1(

input clk,rst,clk\_1M,

input[4:0] clk\_cnt,

input[20:0] count\_1us,

inout dht11,

output reg [31:0] data\_valid,

output reg us\_clear,TH

);

//define parameter

parameter POWER\_ON\_NUM = 1000\_000;

parameter S\_POWER\_ON = 3'd0;

parameter S\_LOW\_20MS = 3'd1;

parameter S\_HIGH\_20US = 3'd2;

parameter S\_LOW\_83US = 3'd3;

parameter S\_HIGH\_87US = 3'd4;

parameter S\_SEND\_DATA = 3'd5;

parameter S\_DELAY = 3'd6;

reg[2:0] ps;

reg[2:0] ns;

reg[5:0] data\_count;

reg[39:0] data\_temp;

reg us\_cnt\_clr;

reg state;

reg dht\_buffer;

reg dht\_d0;

reg dht\_d1;

wire dht\_podge; //data posedge

wire dht\_nedge; //data negedge

assign dht11 = dht\_buffer;

assign dht\_podge = ~dht\_d1 & dht\_d0; // catch posedge

assign dht\_nedge = dht\_d1 & (~dht\_d0); // catch negedge

always @ (posedge clk\_1M or negedge rst) begin

if (!rst)

ps <= S\_POWER\_ON;

else

ps <= ns;

end

always @ (posedge clk\_1M or negedge rst) begin

if(!rst)

begin

ns <= S\_POWER\_ON;

dht\_buffer <= 1'bz;

state <= 1'b0;

us\_clear <= 1'b0;

data\_temp <= 40'd0;

data\_count <= 6'd0;

end

else

begin

case (ps)

S\_POWER\_ON : //wait

begin

if(count\_1us < POWER\_ON\_NUM)

begin

dht\_buffer <= 1'bz;

us\_clear <= 1'b0;

end

else

begin

ns <= S\_LOW\_20MS;

us\_clear <= 1'b1;

end

end

S\_LOW\_20MS: // send 20 ms

begin

if(count\_1us < 20000)

begin

dht\_buffer <= 1'b0;

us\_clear <= 1'b0;

end

else

begin

ns <= S\_HIGH\_20US;

dht\_buffer <= 1'bz;

us\_clear <= 1'b1;

end

end

S\_HIGH\_20US: // Hign 20 us

begin

if (count\_1us < 20)

begin

us\_clear <= 1'b0;

if(dht\_nedge)

begin

ns <= S\_LOW\_83US;

us\_clear <= 1'b1;

end

end

else

ns <= S\_DELAY;

end

S\_LOW\_83US:

begin

if(dht\_podge)

ns <= S\_HIGH\_87US;

end

S\_HIGH\_87US: // ready to receive data signal

begin

if(dht\_nedge)

begin

ns <= S\_SEND\_DATA;

us\_clear <= 1'b1;

end

else

begin

data\_count <= 6'd0;

data\_temp <= 40'd0;

state <= 1'b0;

end

end

S\_SEND\_DATA: // have 40 bit

begin

case(state)

0: begin

if(dht\_podge)

begin

state <= 1'b1;

us\_clear <= 1'b1;

end

else

us\_clear <= 1'b0;

end

1: begin

if(dht\_nedge)

begin

data\_count <= data\_count + 1'b1;

state <= 1'b0;

s\_clear <= 1'b1;

if(count\_1us < 60)

data\_temp <= {data\_temp[38:0],1'b0}; //0

else

data\_temp <= {data\_temp[38:0],1'b1}; //1

end

else //wait for high end

us\_clear <= 1'b0;

end

endcase

if(data\_count == 40) //check data bit

begin

ns <= S\_DELAY;

if(data\_temp[7:0] == data\_temp[39:32] + data\_temp[31:24] + data\_temp[23:16] + data\_temp[15:8])

data\_valid <= data\_temp[39:8];

end

end

S\_DELAY: // after data received delay 2s

begin

if(count\_1us < 2000\_000)

us\_cnt\_clr <= 1'b0;

else

begin

ns <= S\_LOW\_20MS; // send signal again

us\_cnt\_clr <= 1'b1;

end

end

default: ps <= ps;

endcase

end

end

//edge

always @ (posedge clk\_1M or negedge rst) begin

if (!rst) begin

dht\_d0 <= 1'b1;

dht\_d1 <= 1'b1;

end

else begin

dht\_d0 <= dht11;

dht\_d1 <= dht\_d0;

end

end

always @(data\_valid)

begin

if ((data\_valid[15:8] > 30))

TH = 1;

else

TH=0;

end

endmodule

1. RTL Verilog Design of Digital Thermometer Circuit - Top Level Module

module DigThermo(

input clk,rst,

inout dht11,

output [6:0] Out1,Out2,Out3,Out4,

output TH

);

wire clk\_1M,us\_clear;

wire [20:0] count\_1us;

wire [4:0] clk\_cnt;

wire [7:0] A1, A2,A3,A4;

wire [31:0] data\_valid;

DU1 u1(.clk(clk),.rst(rst),.us\_clear(us\_clear),.clk\_cnt(clk\_cnt),.count\_1us(count\_1us),.clk\_1M(clk\_1M));

CU1 u2( .clk(clk),.rst(rst),.us\_clear(us\_clear),.clk\_cnt(clk\_cnt),.count\_1us(count\_1us),.clk\_1M(clk\_1M),.dht11(dht11),.data\_valid(data\_valid),.TH(TH));

bintobcd u3( .bin(data\_valid[31:24]), .bcdout(A1));

bintobcd u5( .bin(data\_valid[15:8]), .bcdout(A3));

Seven\_segment\_decoder u7(.bcdin(A1[7:4]),.seven\_seg(Out1));

Seven\_segment\_decoder u8(.bcdin(A1[3:0]),.seven\_seg(Out2));

Seven\_segment\_decoder u9(.bcdin(A3[7:4]),.seven\_seg(Out3));

Seven\_segment\_decoder u10(.bcdin(A3[3:0]),.seven\_seg(Out4));

endmodule

4. Verilog Design of 7-segment LED Display

module Seven\_segment\_decoder(

input [3:0] bcdin,

output reg [6:0] seven\_seg

);

//common cathode

always @(bcdin)

begin

case (bcdin)

4'b0000 : begin seven\_seg = 7'b1111110; end

4'b0001 : begin seven\_seg = 7'b0110000; end

4'b0010 : begin seven\_seg = 7'b1101101; end

4'b0011 : begin seven\_seg = 7'b1111001; end

4'b0100 : begin seven\_seg = 7'b0110011; end

4'b0101 : begin seven\_seg = 7'b1011011; end

4'b0110 : begin seven\_seg = 7'b1011111; end

4'b0111 : begin seven\_seg = 7'b1110000; end

4'b1000 : begin seven\_seg = 7'b1111111; end

4'b1001 : begin seven\_seg = 7'b1110011; end

default : begin seven\_seg = 7'b0000000; end

endcase

end

endmodule

module bintobcd(

input [7:0] bin,

output reg[11:0] bcdout

);

reg [3:0]i;

always @(bin)

begin

bcdout = 0; //initialize bcd to zero.

for (i = 0; i < 8; i = i+1) //run for 8 iterations

begin

bcdout = {bcdout[11:0],bin[7-i]}; //concatenation

//if a hex digit of 'bcd' is more than 4, add 3 to it.

if(i < 7 && bcdout[3:0] > 4)

bcdout[3:0] = bcdout[3:0] + 3;

if(i < 7 && bcdout[7:4] > 4)

bcdout[7:4] = bcdout[7:4] + 3;

if(i < 7 && bcdout[11:8] > 4)

bcdout[11:8] = bcdout[11:8] + 3;

end

end

endmodule

1. Verilog code of DHT11 (Simplified Verilog Code for Verification Process)

module dht11(

input clk,

input rst\_n,

input dht11,

output reg [7:0] Out1,Out2,Out3,Out4,

output reg TH,TL,

output reg [31:0] data\_valid

);

//reg define

reg[5:0] data\_count;

reg[39:0] data\_temp;

reg [3:0] i;

reg count\_finish;

// state machine

always @ (posedge clk or negedge rst\_n)

begin

if(!rst\_n)

begin

data\_temp <= 40'd0;

data\_count <= 6'd0;

end

else

begin

data\_count <= data\_count + 1'b1;

case(dht11)

0: data\_temp <= {data\_temp[38:0],1'b0} ;

1: data\_temp <= {data\_temp[38:0],1'b1};

endcase

if(data\_count == 40) //check data bit

begin

count\_finish=1;

data\_valid <= data\_temp[39:8];

end

end

end

always@(data\_valid or count\_finish)

begin

TH=0; TL=0;

if(count\_finish==1)

begin

for (i = 0; i < 8; i = i+1)

begin

Out1[i] = data\_valid[24+i];

Out2[i] = data\_valid[16+i];

Out3[i] = data\_valid[8+i];

Out4[i] = data\_valid[i];

end

if ((data\_valid[15:8] > 35))

TH = 1;

else

TL = 1;

end

end

endmodule

1. Testbench

module testbench;

reg clk, rst\_n;

reg dht11;

wire[7:0] Out1,Out2,Out3,Out4;

wire TH,TL;

wire [31:0] data\_valid;

dht11 dut(.clk(clk), .rst\_n(rst\_n), .dht11(dht11),.Out1(Out1), .Out2(Out2), .Out3(Out3), .Out4(Out4), .TH(TH), .TL(TL),.data\_valid(data\_valid));

initial begin

clk = 0;

$monitor ("time: ", $time,

"clk=%b, rst=%b, dht11=%b, Out1=%b, Out2=%b, Out3=%b, Out4=%b, TH=%b, TL=%b, data\_valid=%b\n;", clk, rst\_n, dht11, Out1, Out2, Out3, Out4, TH, TL, data\_valid);

$dumpfile("dump.vcd");

$dumpvars;

end

always #5 clk = ~clk;

initial begin

#10 rst\_n=0; dht11=1;

#10 rst\_n=1; dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#10 dht11=1;

#10 dht11=0;

#200 $finish;

end

endmodule

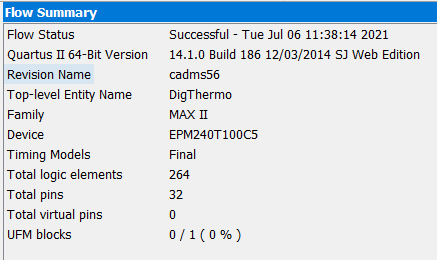
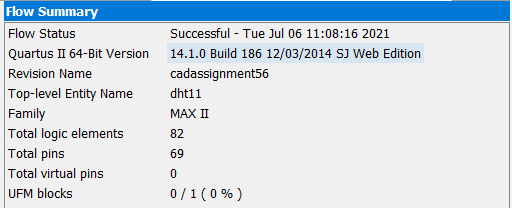
 

Figure 4: The compilation report of DHT11 Verilog code (Simplified Verilog)

Figure 3: The compilation report of Digital Thermometer - Top Level Module

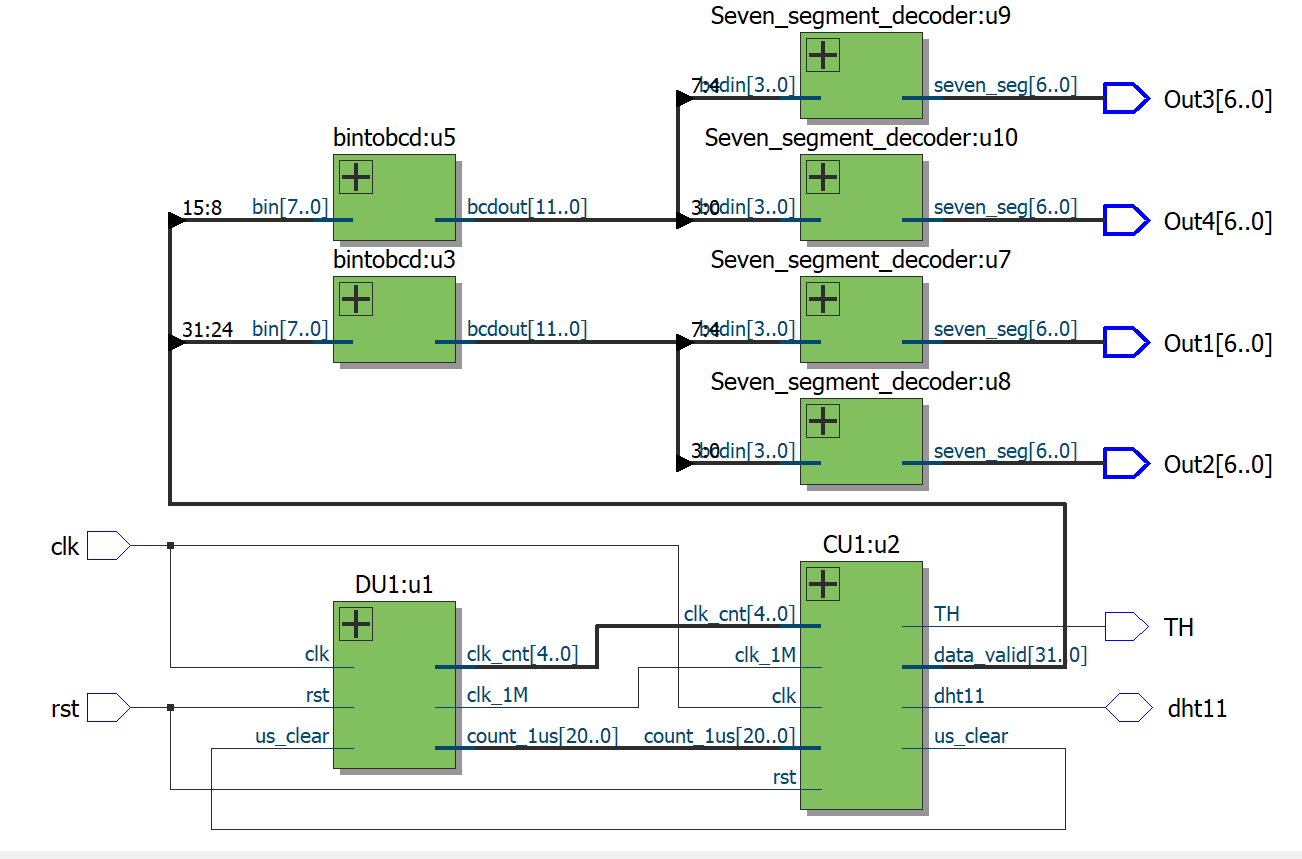


Figure 5: The schematic design by combining all the DU, CU and 7-segmemt LED display

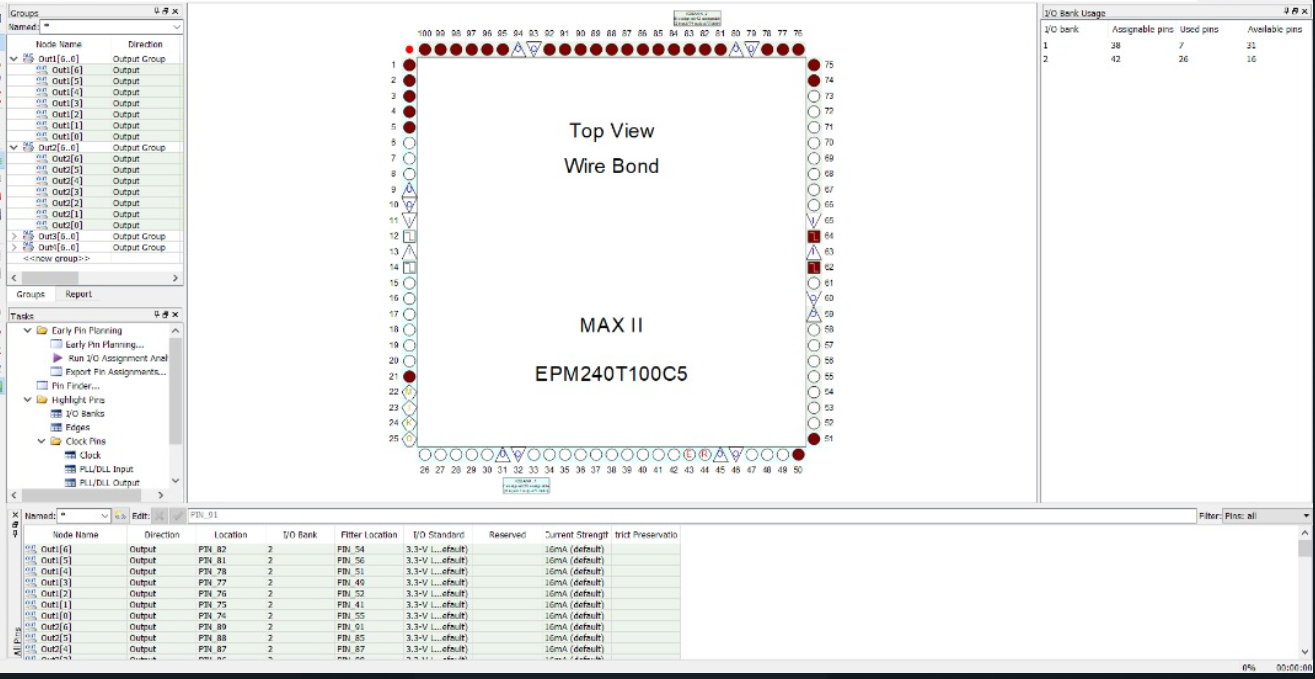
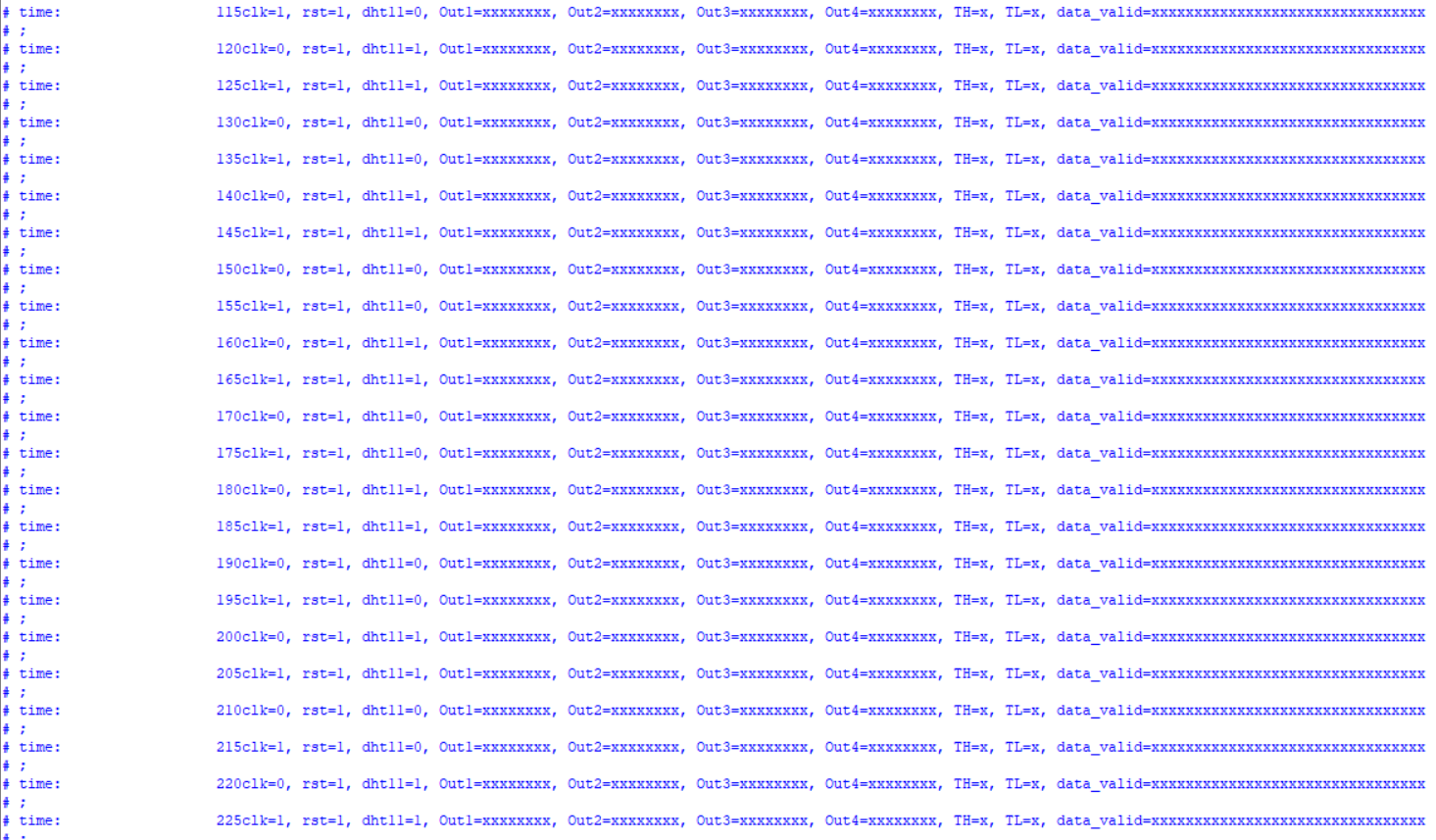
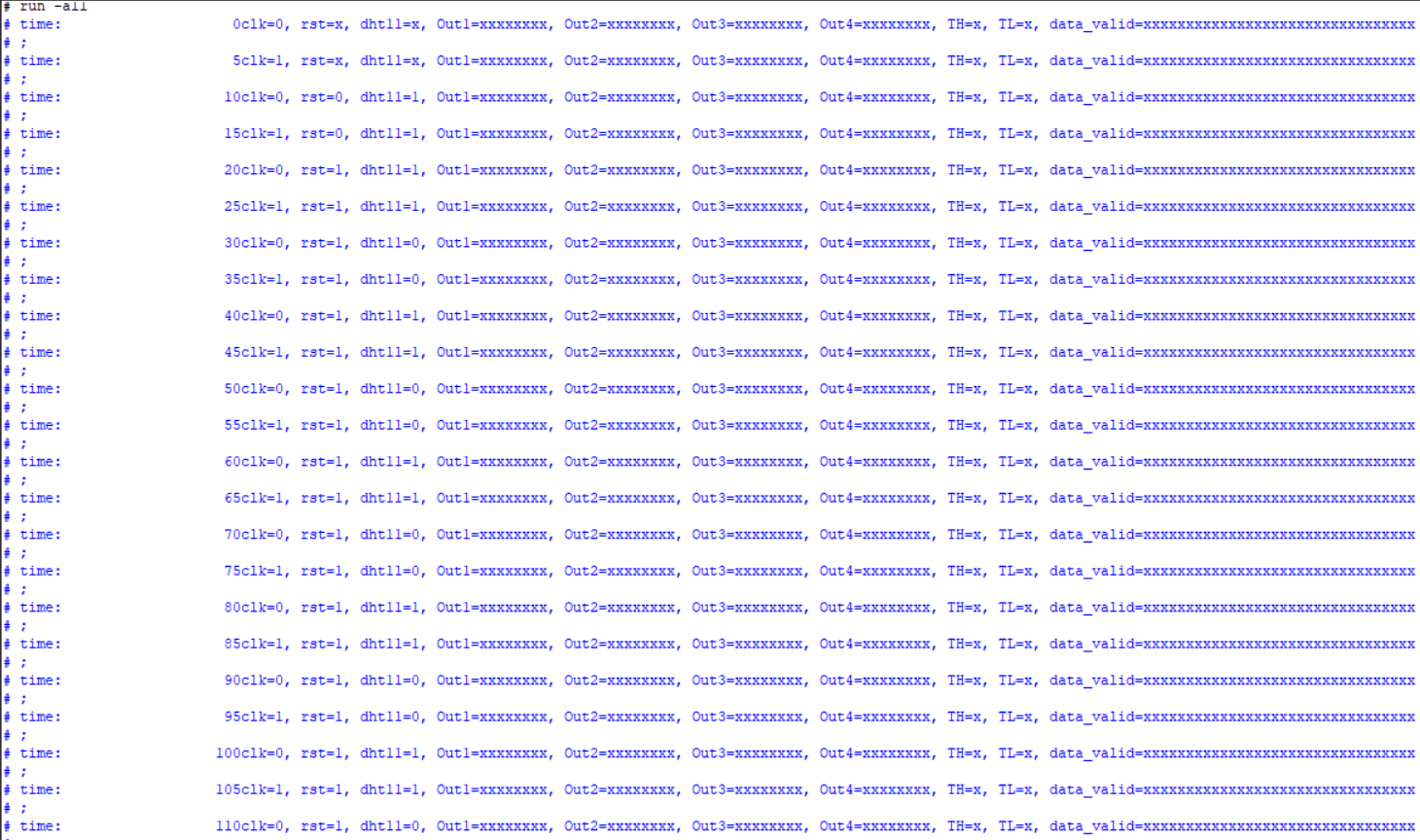
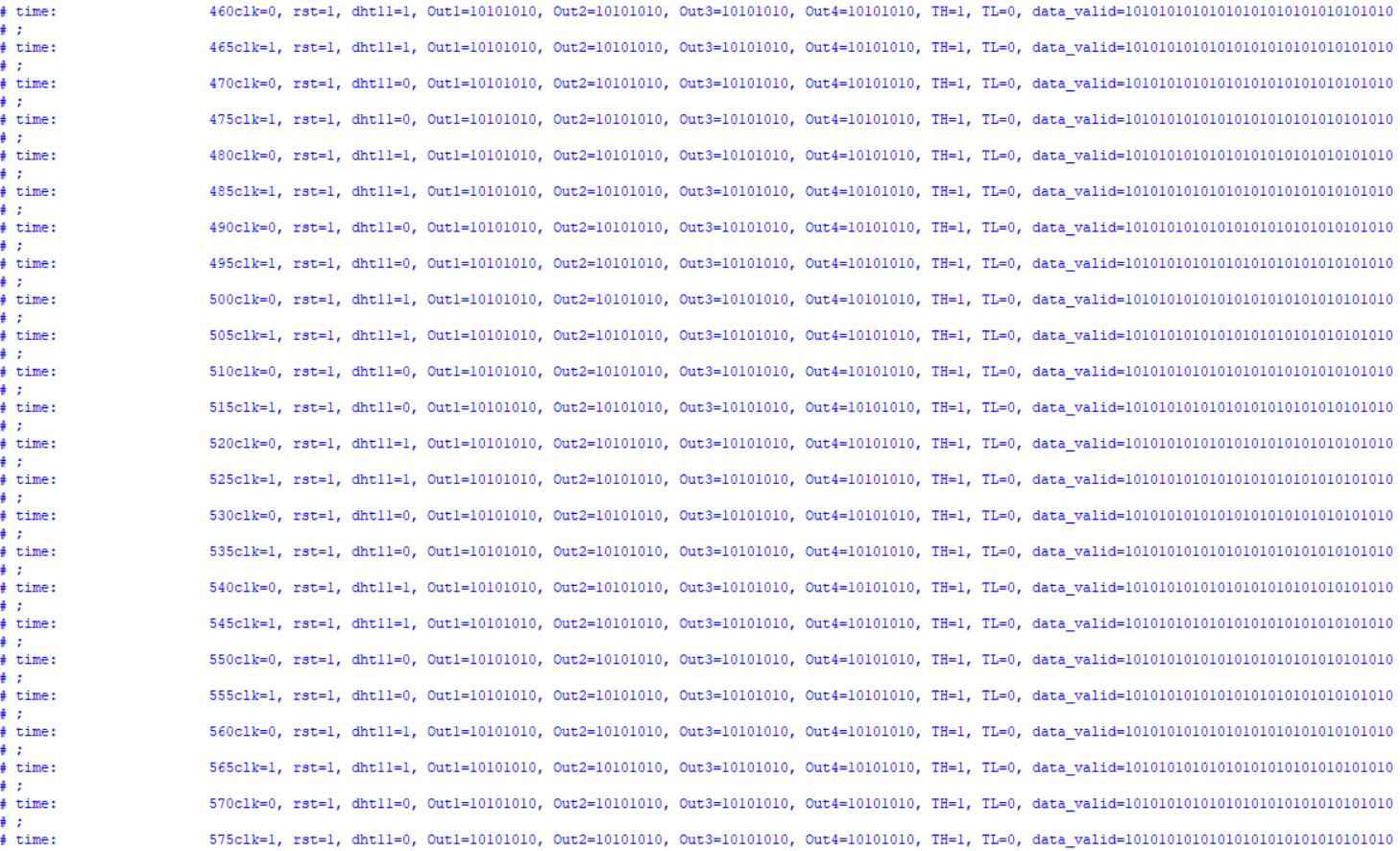
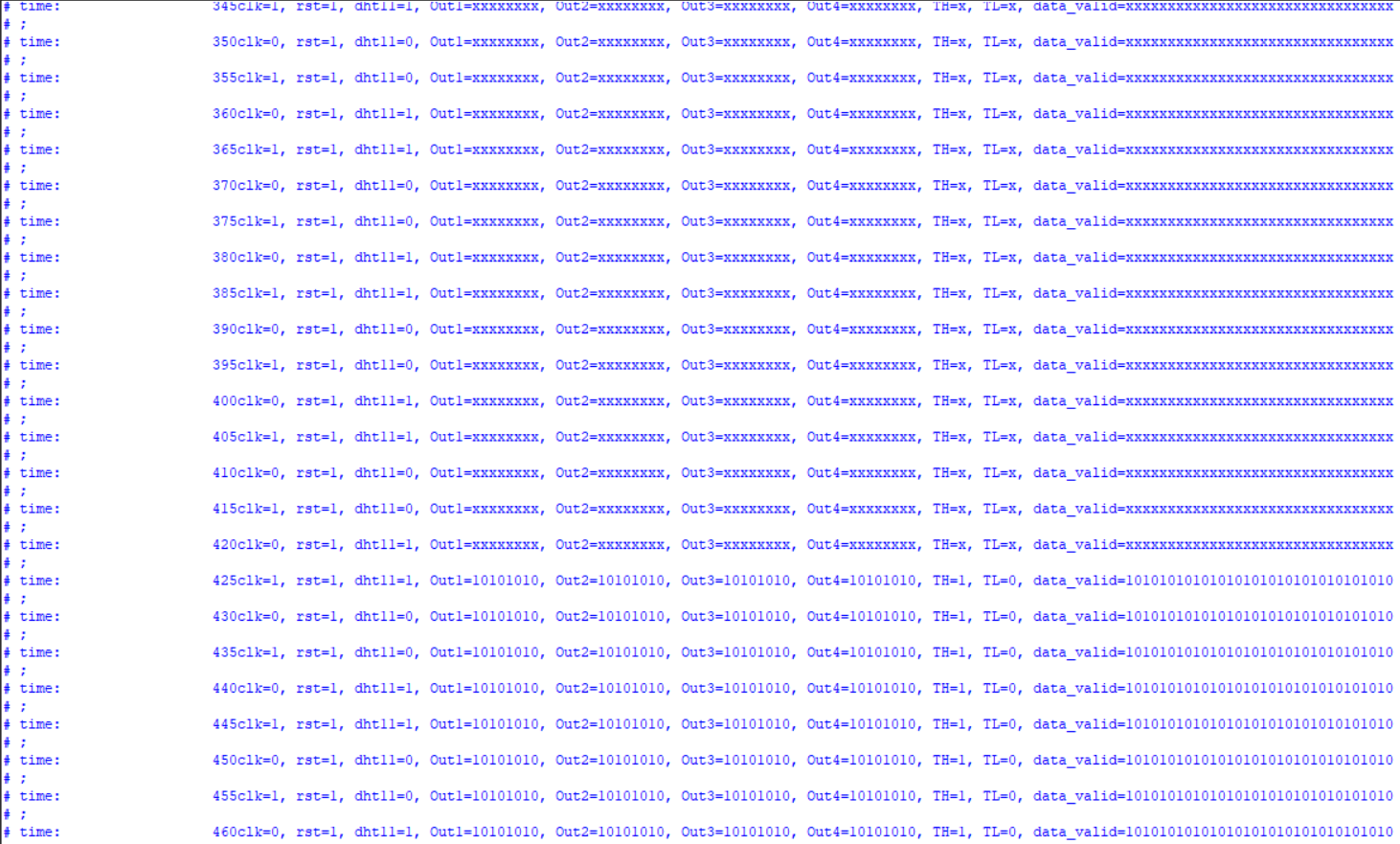
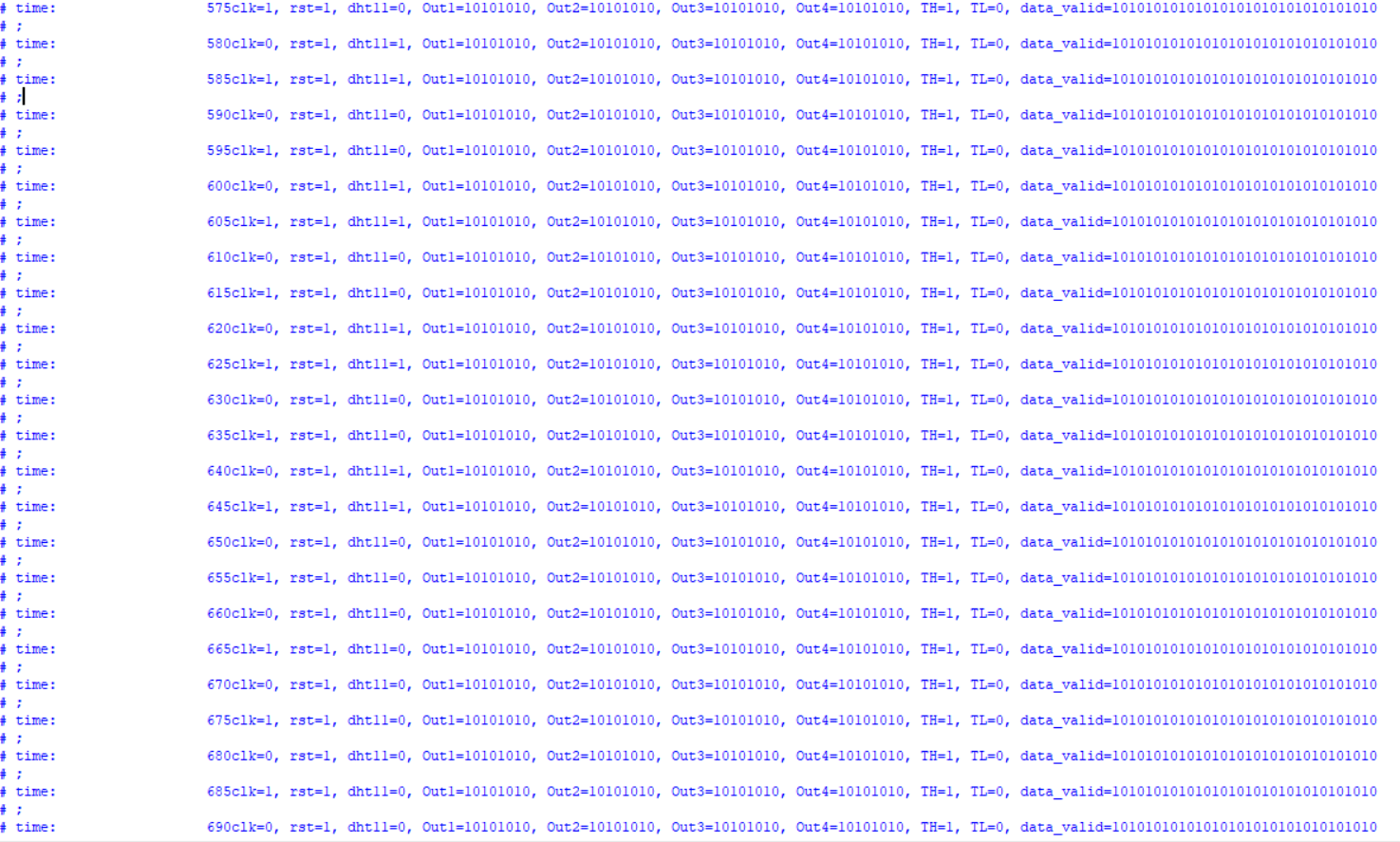


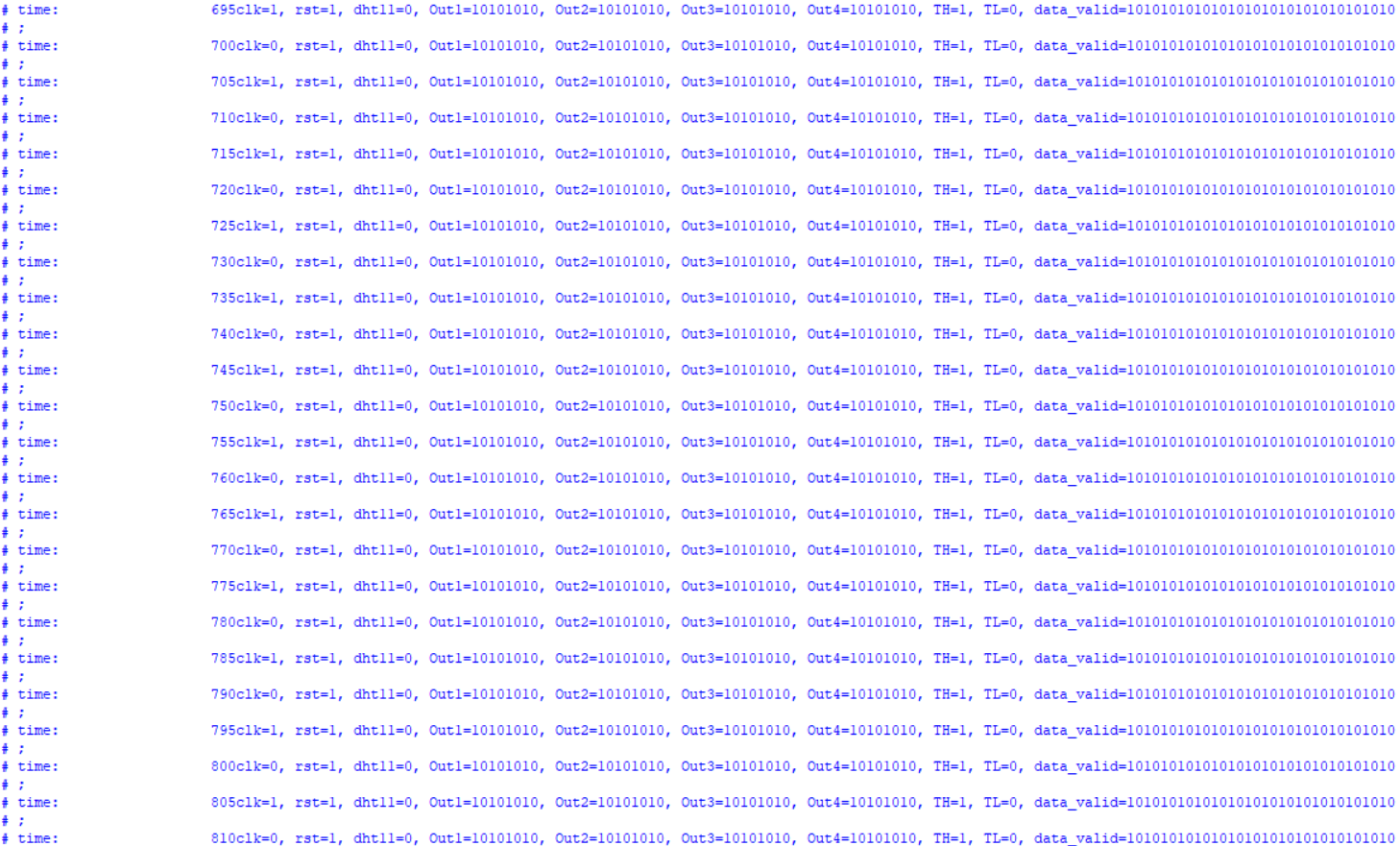
Figure 6: The pin planner of DigThermo.











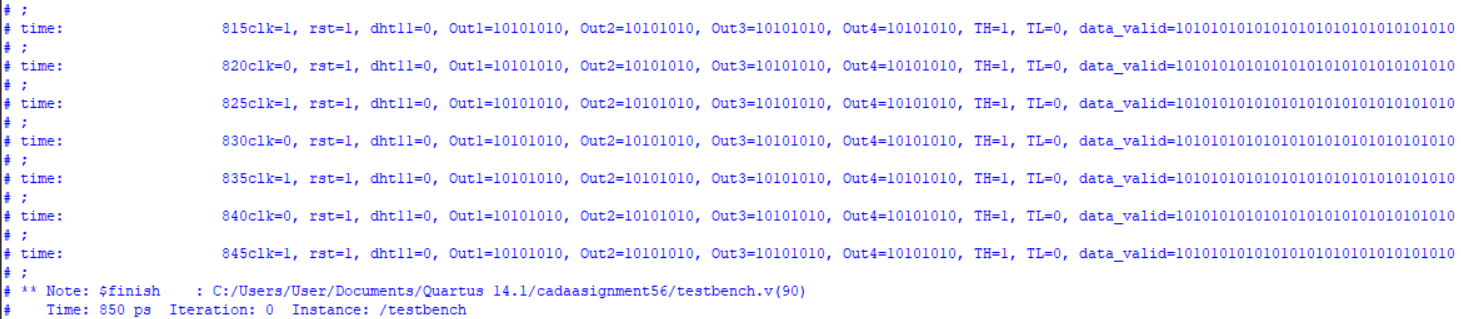


Figure 7: The simulation result of DHT11

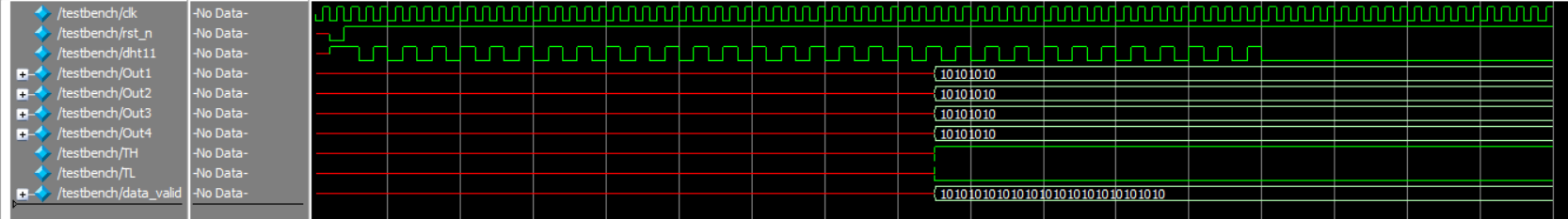


Figure 8: The RTL Simulation Waveform of DHT11

As seen in the Figure 7, the when dht11 sensor start to read the input data, no output shown after 430ps. After 430ps, TH give high output means the green LED will light up and give a signal to switch on the air conditioner.

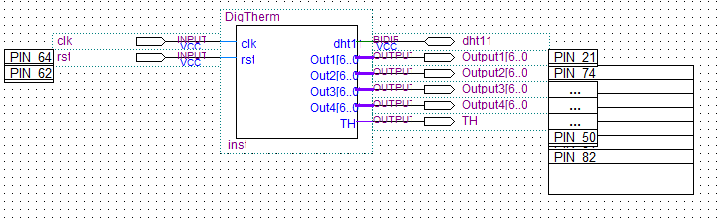


Figure 9: The schematic diagram of digital thermometer

**Part II: Hardware Demonstration**

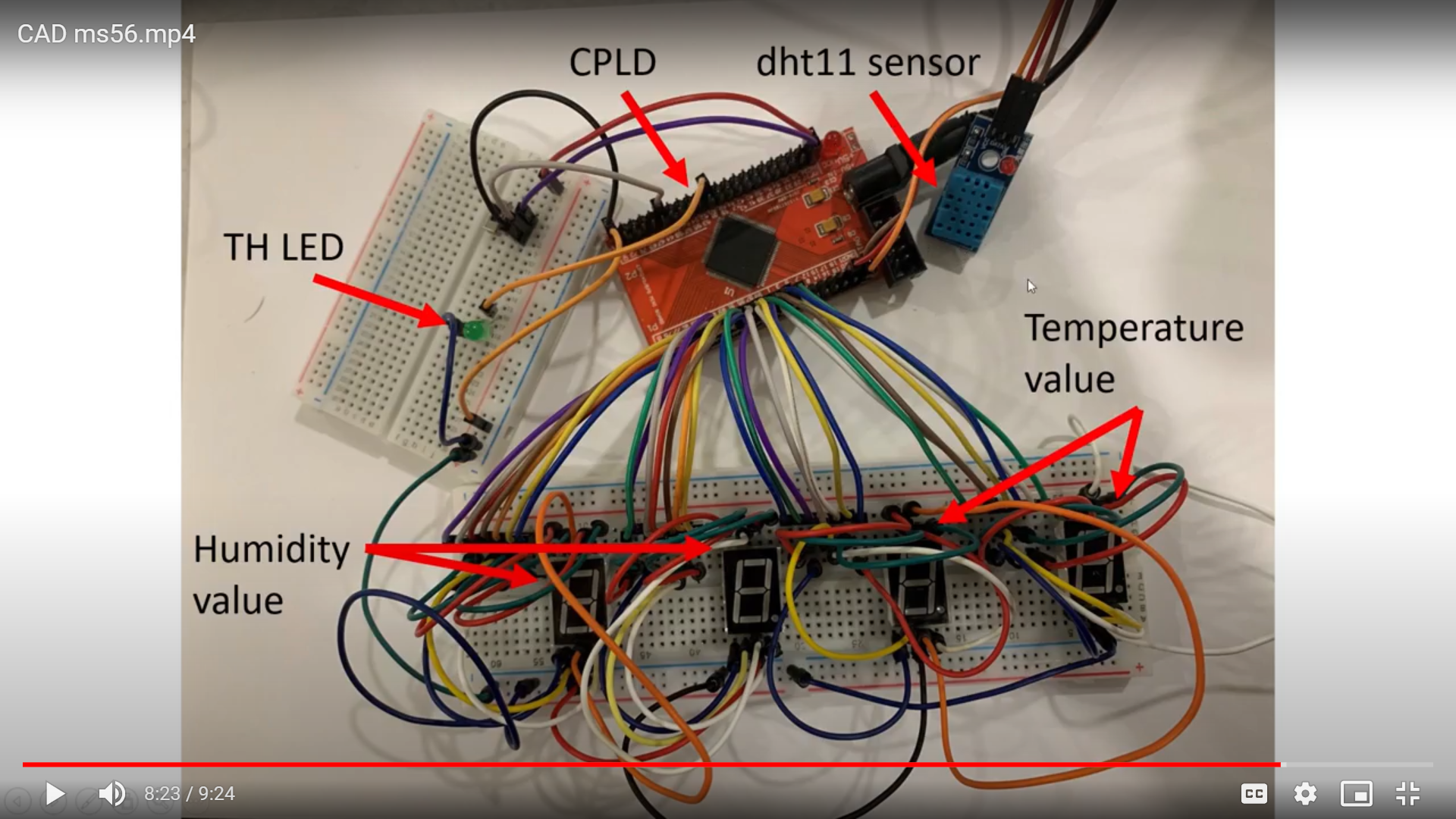


Figure 10: The Hardware Connection DHT11 sensor on CPLD

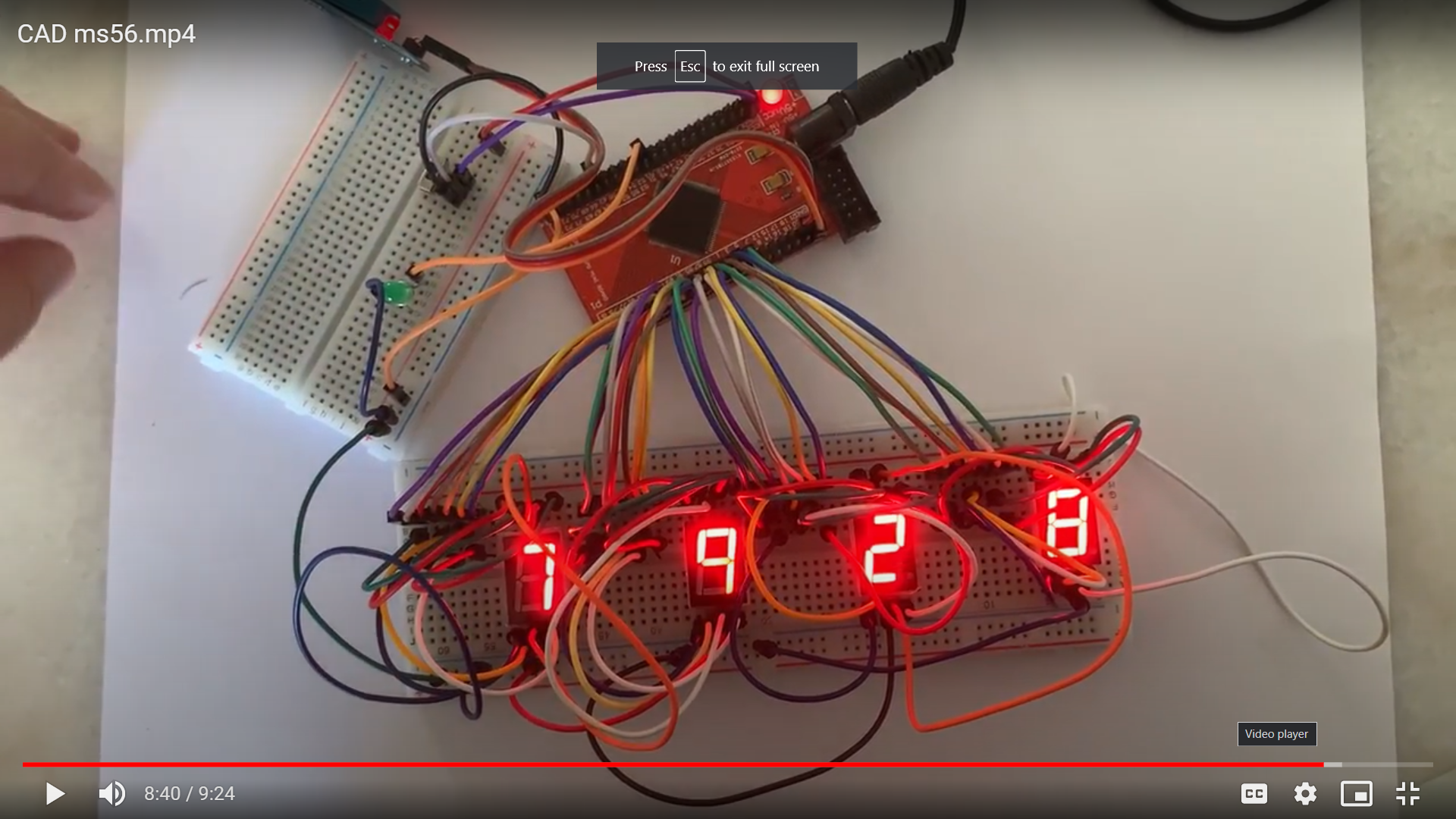


Figure 11: The Result of Hardware Connection DHT11 sensor on CPLD

Figure 11 show the result of hardware connection dht11 sensor. As seen in the 4 7-segment LED display, the humidity level is 79% and the temperature is 28°C. The green LED is not lighting up because the temperature is lower than 30°C. This indicates that there is no signal to switch on air-conditioner.

Hardware Demonstration Video Link: https://youtu.be/lAtVJhyK9vA

References

1. K. Mahmud, M. S. Alam and R. Ghosh, "Design of digital thermometer based on PIC16F77A single chip microcontroller," 2013 3rd International Conference on Consumer Electronics, Communications and Networks, 2013, pp. 246-249, doi: 10.1109/CECNet.2013.6703317.
2. Gay W. (2018) DHT11 Sensor. In: Advanced Raspberry Pi. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3948-3\_22