Laporan Tugas Besar
Desain FPGA dan SoC 2025

Desain FPGA dan SoC
Kelompok : Kelompok 3
Nama-NIM Anggota 1 : Muhammad Fathir Qinthara - 1102223006
Nama-NIM Anggota 2 : Vincentius Artyanta Mahesa -1102223079
Nama-NIM Anggota 3 : Ivan Horas Hamonangan Simaremare 1102223105

Judul

Sistem Deteksi Suhu dan Kelembapan Berbasis FPGA dengan tampilan Data pada 7 Segment Display. Judul ini menggambarkan perancangan sebuah sistem digital berbasis FPGA yang berfungsi untuk membaca, mengolah, dan menampilkan data suhu serta kelembapan lingkungan secara real-time melalui media tampilan 7-segment display.

Deskripsi

Tugas besar ini bertujuan untuk merancang dan mengimplementasikan sistem pendeteksi suhu dan kelembapan berbasis FPGA. Sistem bekerja dengan menerima data dari sensor suhu dan kelembapan, kemudian data tersebut diproses menggunakan logika digital yang diimplementasikan pada FPGA. Hasil pengolahan data selanjutnya ditampilkan dalam bentuk nilai numerik pada 7-segment display

Perancangan mencakup pembacaan sensor dari modul DHT11, pengolahan data, serta pengendalian tampilan yang terintegrasi antara hardware sensor dan hardware FPGA.

Fungsi

- Mendeteksi dan membaca nilai suhu serta kelembapan dari sensor.
- Mengolah data sensor menggunakan logika digital berbasis Quartus dan implementasi ke hardware FPGA.
- Menampilkan hasil pengukuran suhu dan kelembapan pada 7-segment display.

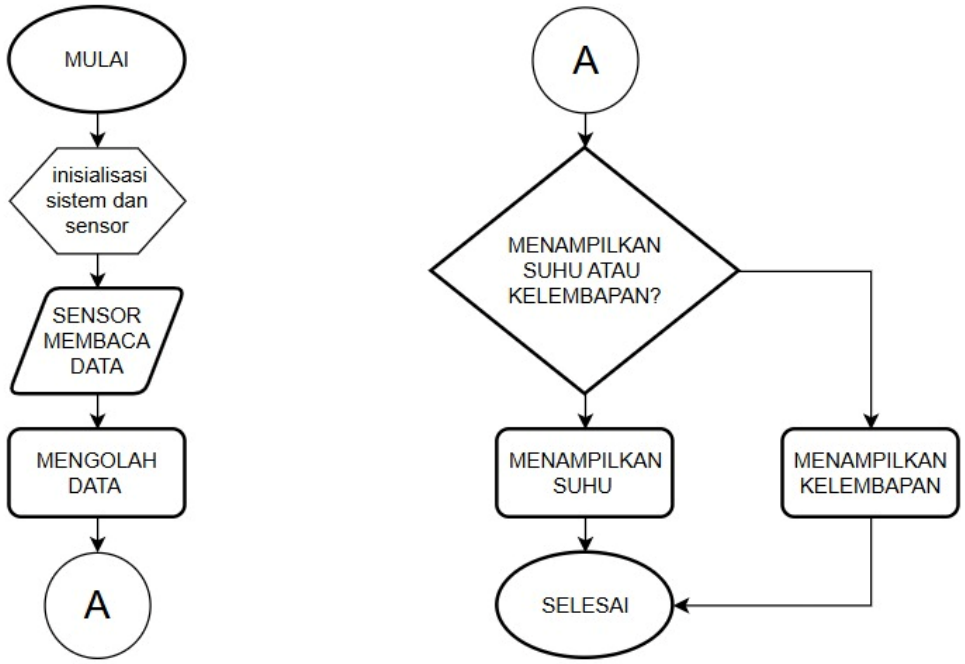
Fitur dan Spesifikasi

Contoh alat	Contoh Fitur
Pengukur Suhu dan Kelembapan Ruangan	Mengukur nilai suhu dan kelembapan ruangan dan menampilkan nilainya secara digital

Contoh spesifikasi:

- Sistem mampu mengukur suhu dan kelembapan ruangan secara digital
- Output nilai ditampilkan melalui 7-segment display dan indikator LED
- Tampilan suhu dan kelembapan dapat ditampilkan secara bergantian selama 5 detik

Cara Penggunaan

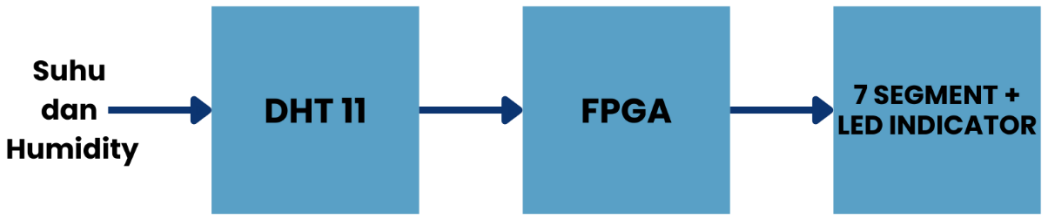


Flowchart diawali dengan melakukan setup di aplikasi Quartus dan mulai membuat code program sistem. Setelah proses inisialisasi selesai, sensor mulai dapat membaca data suhu dan kelembapan dari lingkungan. Data yang diperoleh selanjutnya diolah oleh sistem sebelum ditampilkan. Pada tahap mengolah data, proses tidak hanya mencakup pengolahan data saat sistem berjalan, tetapi juga meliputi tahapan perancangan dan implementasi sistem pada FPGA menggunakan perangkat lunak Quartus. Tahapan implementasi yang dilakukan adalah sebagai berikut:

1. Menjalankan program Verilog HDL
2. Melakukan inisialisasi dan pengaturan pin (Pin Planner)
3. Mengunggah (upload) kode ke FPGA

Selanjutnya, alur proses berpindah ke titik A untuk menentukan jenis data yang akan ditampilkan. Apabila sistem memilih suhu, maka nilai suhu akan ditampilkan pada output. Sebaliknya, jika sistem memilih kelembapan, maka nilai kelembapan akan ditampilkan. Setelah salah satu data ditampilkan, proses diakhiri pada tahap Selesai.

Blok Diagram

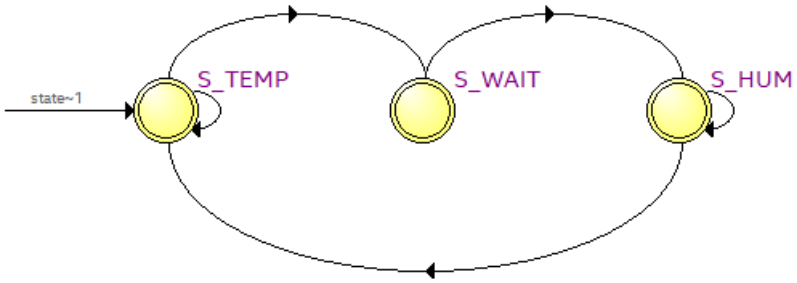


- Sensor DHT11 berfungsi sebagai perangkat input yang membaca data suhu dan kelembapan dari lingkungan. Data yang dihasilkan oleh sensor ini dikirimkan dalam bentuk sinyal digital menuju FPGA.

- FPGA berperan sebagai proses atau pengendali utama. Pada bagian ini, data dari DHT11 diproses menggunakan rangkaian logika yang dirancang dengan Verilog HDL.

- Hasil pengolahan data dari FPGA selanjutnya ditampilkan pada seven segment display untuk menunjukkan nilai suhu atau kelembapan dalam bentuk angka. Selain itu, LED indikator digunakan sebagai penanda mode tampilan, di mana LED menunjukkan apakah sistem sedang menampilkan suhu atau kelembapan..

FSM



Finite State Machine (FSM) digunakan untuk mengatur pergantian mode tampilan suhu dan kelembapan secara otomatis berdasarkan waktu simulasi. FSM bekerja secara sinkron terhadap clock lambat (slow clock) dan akan berpindah state ketika kondisi transisi terpenuhi.

1. **State S_TEMP (Temperature Mode)**
   - Pada state ini, sistem menampilkan nilai suhu (25°C) pada seven-segment display.
   - Output aktif: LEDR0 = 1 (indikator suhu menyala)

     LEDR1 = 0
   - Kondisi bertahan: FSM tetap berada di S_TEMP selama counter waktu belum mencapai nilai SIM_SWITCH.
   - Transisi: Jika counter mencapai batas waktu, FSM berpindah ke state S_WAIT.

2. **State S_WAIT (State Transisi)**
   - State ini berfungsi sebagai state perantara antara mode suhu dan mode kelembapan.
   - Transisi: FSM langsung berpindah ke state S_HUM pada siklus clock berikutnya.

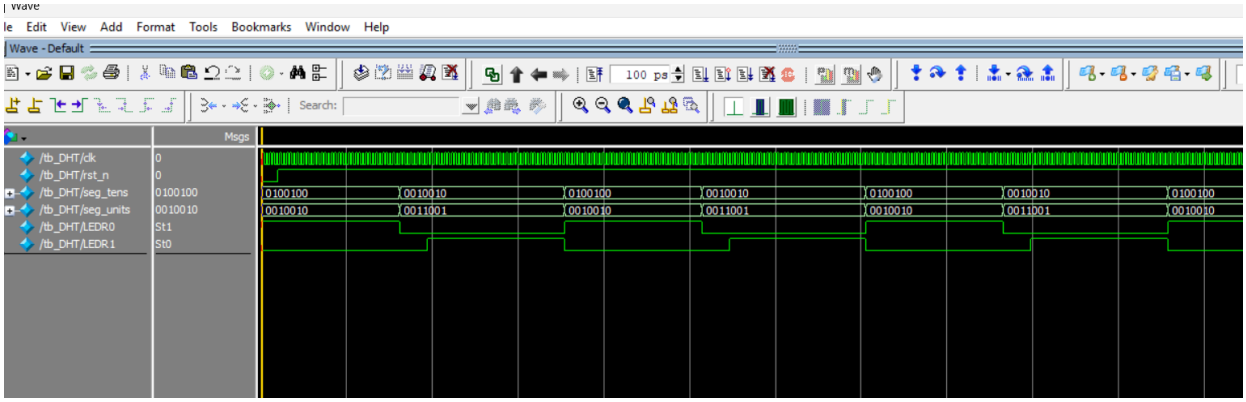
3. **State S_HUM (Humidity Mode)**
   - Pada state ini, sistem menampilkan nilai kelembapan (54%) pada seven-segment display
   - Output aktif: LEDR0 = 0

     LEDR1 = 1 (indikator kelembapan menyala)
   - Kondisi bertahan: FSM tetap berada di S_HUM selama counter waktu belum mencapai nilai SIM_SWITCH
   - Transisi: Jika waktu tercapai, FSM kembali ke state S_TEMP.

Hasil simulasi dan Analisis

1. Skenario pengujian yang dilakukan adalah sebagai berikut:
   - Sistem diberikan sinyal clock dan reset aktif-low (rst_n).
   - Setelah reset dilepas, FSM memulai operasi pada mode suhu.
   - Sistem menampilkan nilai suhu 25°C pada seven segment dan menyalakan LEDR0.
   - Setelah waktu simulasi tertentu (diasumsikan 5 detik), FSM berpindah ke mode kelembapan.
   - Sistem menampilkan nilai kelembapan 54% pada seven segment dan menyalakan LEDR1.
   - Proses berlangsung secara berulang sesuai transisi FSM.

2. Simulasi yang diperoleh



   - Sinyal clk berosilasi secara stabil sebagai clock utama sistem.
   - Sinyal rst_n aktif low pada awal simulasi dan dilepas untuk memulai operasi sistem atau untuk melakukan reset.
   - Keluaran seven segment (seg_tens dan seg_units) menampilkan angka **25** pada mode suhu dan **54** pada mode kelembapan.
   - LED indikator bekerja secara bergantian: LEDR0 menyala saat mode suhu aktif dan LEDR1 menyala saat mode kelembapan aktif.
   - Pergantian tampilan terjadi secara periodik sesuai dengan waktu simulasi yang telah ditentukan

3. Analisis Hasil Simulasi

   Berdasarkan waveform simulasi, setelah sinyal reset (rst_n) dilepas, sistem bekerja secara normal. Saat kondisi pertama, seven segment akan menampilkan nilai 25, ditunjukkan oleh seg_tens = 0100100 (angka 2) dan seg_units = 0010010 (angka 5). Pada kondisi ini, LEDR0 bernilai 1 (menyala) dan LEDR1 bernilai 0 (mati).

   Setelah interval waktu simulasi tercapai, tampilan berubah menjadi nilai 54, dengan seg_tens = 0010010 (angka 5) dan seg_units = 0011001 (angka 4). Bersamaan dengan perubahan tampilan di seven segment, LEDR1 bernilai 1 (menyala) dan LEDR0 bernilai 0 (mati).

Lampiran (Kode Verilog)

1. Kode Verilog Simulasi

- **Top Module.v**

```
module DHT (
    input  wire clk,
    input  wire rst_n,
    output wire [6:0] seg_tens,
```

```
    output wire [6:0] seg_units,
    output wire LEDR0,
    output wire LEDR1
);

    // ==========================
    // PARAMETER SIMULASI CEPAT
    // ==========================
    parameter SIM_SWITCH = 5;   // anggap 5 detik (SIMULASI)

    // ==========================
    // FSM STATE DECLARATION
    // ==========================
    parameter S_TEMP = 2'd0,
              S_WAIT = 2'd1,
              S_HUM  = 2'd2;

    reg [1:0] state, next_state;

    // ==========================
    // INTERNAL SIGNAL
    // ==========================
    wire slow_clk;
    reg  [3:0] switch_cnt;

    // ==========================
    // CLOCK DIVIDER (SIMULASI)
    // ==========================
    clock_divider #(.DIV(4)) u_clk (
        .clk_in(clk),
        .rst_n(rst_n),
        .clk_out(slow_clk)
    );

    // ==========================
    // FSM STATE REGISTER
    // ==========================
    always @(posedge slow_clk or negedge rst_n) begin
        if (!rst_n)
            state <= S_TEMP;
        else
            state <= next_state;
    end

    // ==========================
    // FSM NEXT STATE LOGIC
    // (INI YANG DIBACA QUARTUS)
    // ==========================
    always @(*) begin
        next_state = state;
        case (state)
            S_TEMP: if (switch_cnt == SIM_SWITCH-1) next_state = S_WAIT;
            S_WAIT:                                 next_state = S_HUM;
            S_HUM : if (switch_cnt == SIM_SWITCH-1) next_state = S_TEMP;
            default:                                next_state = S_TEMP;
        endcase
    end
```

```
// ==========================
// FSM OUTPUT + COUNTER
// ==========================
always @(posedge slow_clk or negedge rst_n) begin
    if (!rst_n) begin
        switch_cnt <= 0;
    end else begin
        if (state != next_state)
            switch_cnt <= 0;
        else
            switch_cnt <= switch_cnt + 1;
    end
end

// ==========================
// LED INDIKATOR MODE
// ==========================
assign LEDR0 = (state == S_TEMP); // suhu
assign LEDR1 = (state == S_HUM);  // humidity

// ==========================
// NILAI DISPLAY FIX
// ==========================
wire [7:0] display_value;
assign display_value = (state == S_TEMP) ? 8'd25 : 8'd54;

// ==========================
// BCD CONVERTER
// ==========================
wire [3:0] tens;
wire [3:0] units;

bcd_converter u_bcd (
    .bin(display_value),
    .tens(tens),
    .units(units)
);

// ==========================
// SEVEN SEGMENT
// ==========================
seven_segment_decoder u_seg_tens (
    .digit(tens),
    .seg(seg_tens)
);

seven_segment_decoder u_seg_units (
    .digit(units),
    .seg(seg_units)
);

Endmodule
```

- **Bcd_Converter.v**

```
module bcd_converter (
    input  wire [7:0] bin,
    output wire [3:0] tens,
```

```
    output wire [3:0] units
);

    assign tens  = bin / 10;
    assign units = bin % 10;

endmodule
```

- **Clock_Divider.v**

```
module clock_divider #(
    parameter DIV = 4
)(
    input  wire clk_in,
    input  wire rst_n,
    output reg  clk_out
);

    reg [$clog2(DIV)-1:0] cnt;

    always @(posedge clk_in or negedge rst_n) begin
        if (!rst_n) begin
            cnt <= 0;
            clk_out <= 0;
        end else if (cnt == DIV-1) begin
            cnt <= 0;
            clk_out <= ~clk_out;
        end else begin
            cnt <= cnt + 1;
        end
    end

endmodule
```

- **Sensor_Simulator.v**

```
module sensor_simulator (
    input  wire clk,
    input  wire rst_n,
    output reg [7:0] temp,
    output reg [7:0] hum
);

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            temp <= 25;
            hum  <= 60;
        end else begin
            temp <= temp + 1;
            hum  <= hum + 2;
        end
    end
endmodule
```

- **Seven_Segment_Decoder.v**

```
module seven_segment_decoder (
    input  wire [3:0] digit,
```

```
    output reg  [6:0] seg
);

    always @(*) begin
        case (digit)
            4'd0: seg = 7'b1000000;
            4'd1: seg = 7'b1111001;
            4'd2: seg = 7'b0100100;
            4'd3: seg = 7'b0110000;
            4'd4: seg = 7'b0011001;
            4'd5: seg = 7'b0010010;
            4'd6: seg = 7'b0000010;
            4'd7: seg = 7'b1111000;
            4'd8: seg = 7'b0000000;
            4'd9: seg = 7'b0010000;
            default: seg = 7'b1111111;
        endcase
    end

endmodule
```

- **Test Bench.v**

```
`timescale 1ns/1ps
module tb_DHT;

    reg clk;
    reg rst_n;

    wire [6:0] seg_tens;
    wire [6:0] seg_units;
    wire LEDR0;
    wire LEDR1;

    DHT dut (
        .clk(clk),
        .rst_n(rst_n),
        .seg_tens(seg_tens),
        .seg_units(seg_units),
        .LEDR0(LEDR0),
        .LEDR1(LEDR1)
    );

    // CLOCK 50 MHz
    always #10 clk = ~clk;

    initial begin
        clk = 0;
        rst_n = 0;

        #100;
        rst_n = 1;   // RESET DILEPAS

        #200_000;    // SIMULASI PANJANG
        $stop;
    end

endmodule
```

2. Kode Verilog Implementasi Hardware

```
module dht11_reader (
    input  wire clk,
    input  wire KEY0,          // reset (active low)
    inout  wire dht_data,
    output wire [6:0] seg_tens,
    output wire [6:0] seg_units,
    output wire LEDR0,         // temperature indicator
    output wire LEDR1          // humidity indicator
);

    // ================================
    // DHT11 SIGNALS
    // ================================
    reg [39:0] data_buffer;
    reg [5:0]  bit_index;
    reg [19:0] counter;
    reg [25:0] counter_new_temp;
    reg        previous_data;
    reg [2:0]  state;
    reg        data_direction;
    reg        data_reg;

    reg [7:0] temp;
    reg [7:0] humidity;

    // ================================
    // DISPLAY CONTROL
    // ================================
    reg display_mode;          // 0=temp, 1=humidity
    reg [31:0] display_timer;

    parameter AUTO_SWITCH = 250_000_000; // 5 detik @50MHz

    // ================================
    // DHT11 STATE MACHINE + RESET
    // ================================
    always @(posedge clk) begin
        if (!KEY0) begin
            state <= 0;
            bit_index <= 0;
            counter <= 0;
            counter_new_temp <= 0;
            previous_data <= 0;
            data_direction <= 0;
            temp <= 0;
            humidity <= 0;
        end else begin
            data_reg <= dht_data;

            if (counter_new_temp > 50000000) begin
                state <= 0;
                bit_index <= 0;
                counter <= 0;
                counter_new_temp <= 0;
            end else
```

```
            counter_new_temp <= counter_new_temp + 1;

        case (state)
            0: begin
                data_direction <= 0;
                if (counter > 901000) begin
                    counter <= 0;
                    data_direction <= 1;
                    state <= 1;
                end else
                    counter <= counter + 1;
            end

            1: if (~data_reg) state <= 2;
            2: if ( data_reg) state <= 3;
            3: if (~data_reg) state <= 4;
            4: if ( data_reg) state <= 5;
            5: if (~data_reg) state <= 6;

            6: begin
                if (bit_index < 40) begin
                    if (~data_reg && previous_data) begin
                        data_buffer[39-bit_index] <= (counter > 2500);
                        counter <= 0;
                        bit_index <= bit_index + 1;
                    end
                    if (data_reg)
                        counter <= counter + 1;
                end else begin
                    humidity <= data_buffer[39:32];
                    temp     <= data_buffer[23:16];
                    state <= 0;
                end
                previous_data <= data_reg;
            end
        endcase
    end
end

assign dht_data = (data_direction) ? 1'bz : 1'b0;

// ==============================
// AUTO SWITCH DISPLAY (5 DETIK)
// ==============================
always @(posedge clk) begin
    if (!KEY0) begin
        display_timer <= 0;
        display_mode <= 0;   // default temp
    end else begin
        display_timer <= display_timer + 1;
        if (display_timer >= AUTO_SWITCH) begin
            display_timer <= 0;
            display_mode <= ~display_mode;
        end
    end
end

// ==============================
```

```
    // DISPLAY SELECTION
    // ==============================
    wire [7:0] display_value;
    assign display_value = (display_mode == 1'b0) ? temp : humidity;

    wire [3:0] tens  = display_value / 10;
    wire [3:0] units = display_value % 10;

    seven_seg_decoder seg1 (.digit(tens),  .seg(seg_tens));
    seven_seg_decoder seg0 (.digit(units), .seg(seg_units));

    // ==============================
    // LED MODE INDICATOR
    // ==============================
    assign LEDR0 = (display_mode == 1'b0); // temp
    assign LEDR1 = (display_mode == 1'b1); // humidity

endmodule


// ====================================
// SEVEN SEGMENT DECODER ACTIVE LOW
// ====================================
module seven_seg_decoder (
    input  wire [3:0] digit,
    output reg  [6:0] seg
);
    always @(*) begin
        case (digit)
            4'd0: seg = 7'b1000000;
            4'd1: seg = 7'b1111001;
            4'd2: seg = 7'b0100100;
            4'd3: seg = 7'b0110000;
            4'd4: seg = 7'b0011001;
            4'd5: seg = 7'b0010010;
            4'd6: seg = 7'b0000010;
            4'd7: seg = 7'b1111000;
            4'd8: seg = 7'b0000000;
            4'd9: seg = 7'b0010000;
            default: seg = 7'b1111111;
        endcase
    end
endmodule
```

Link Video Implementasi

[Link Hasil Video Implementasi Kelompok 3]