SMART CAR PARKING SYSTEM VERILOG

Laporan Tugas Besar Aplikasi Sistem Digital

Disusun Oleh:

Hasan Al Banna	1103223142
Dhanendra Nivadirokhman	1103220113
Raffalino Djira Ibrahim	1103223021
Athalah Rafif Irsyach Sarbrani	1103223053
Maulana Hafiz	1103223105

Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom
2024

1. Theory

Seiring dengan pesatnya urbanisasi, permasalahan parkir kendaraan menjadi salah satu tantangan utama yang dihadapi di wilayah perkotaan. Masalah ini sering kali disebabkan oleh dua faktor utama, yaitu: Keterbatasan lahan parkir yang menyebabkan sulitnya menemukan tempat parkir. Kurangnya keamanan dalam pengelolaan parkir, yang dapat mengakibatkan kerugian bagi pengguna. Dalam kehidupan sehari-hari, terutama di area perkotaan dan perkantoran, banyak pengguna kendaraan yang harus menghabiskan waktu lama untuk mencari tempat parkir yang tersedia. Selain itu, sistem parkir konvensional sering kali kurang efisien dan tidak mampu memberikan keamanan yang memadai. Oleh karena itu, diperlukan sebuah solusi yang dapat mengatasi masalah tersebut melalui penerapan teknologi modern. Smart Car Parking System menjadi salah satu pendekatan yang dapat menjawab tantangan ini. Sistem ini dirancang untuk menyediakan pengelolaan parkir yang lebih terstruktur, aman, dan efisien.

Sistem parkir ini dilengkapi dengan deteksi kendaraan menggunakan kamera deteksi gerak yang mengaktifkan keypad untuk memasukkan kata sandi saat kendaraan mendekati pintu masuk parkir. Jika kata sandi yang dimasukkan benar, servo motor akan membuka palang pintu secara otomatis. Setiap slot parkir dilengkapi dengan sensor magnet untuk mendeteksi keberadaan kendaraan, dengan lampu LED yang menyala hijau 3qjika slot kosong dan merah jika slot terisi. Sistem ini memberikan manajemen yang terintegrasi dengan informasi real-time mengenai ketersediaan tempat parkir, sehingga mengurangi waktu pencarian parkir oleh pengguna.

1.1. Algorithm

a. Begin by defining the steps required to achieve the functionality of the system. Langkah langkah yang dibutuhkan untuk dapat memenuhi semua fungsionalitas dari sistem

- Motion_sensor mendeteksi ada tidaknya kendaraan yang berada di lingkup sensor. Pada motion sensor berfungsi untuk mengontrol sinyal keluaran bernama motion_out, yang mengindikasikan keberadaan mobil pada jalur masuk berdasarkan sinyal deteksi car_detected. Ketika sinyal rst aktif tinggi (high), artinya sistem sedang dalam kondisi reset, maka motion_out akan diatur menjadi 1'b0, yang menunjukkan bahwa tidak ada mobil yang

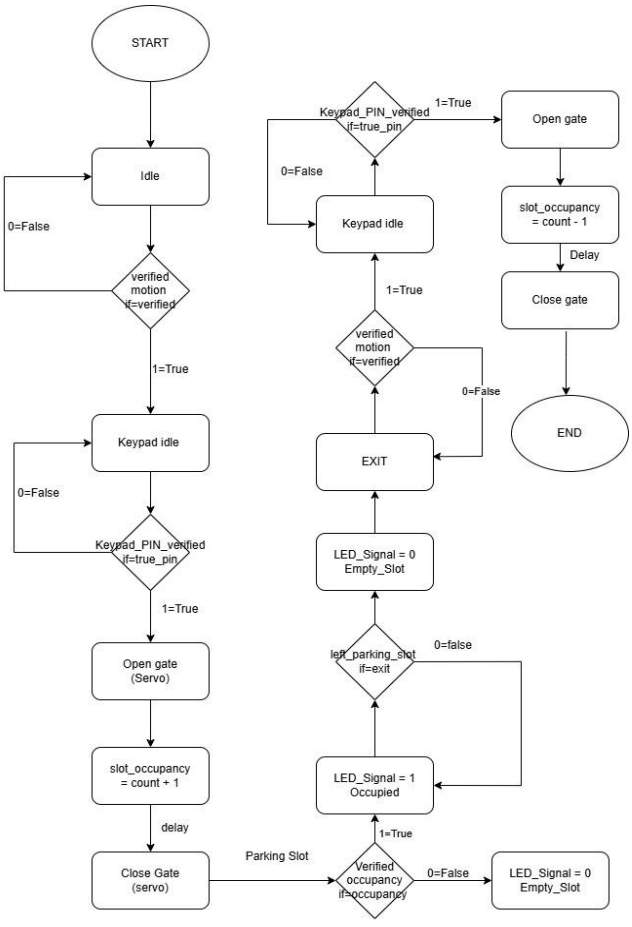
terdeteksi atau sinyal jalur masuk dalam kondisi nonaktif. Namun, ketika sinyal rst tidak aktif (logika rendah), maka nilai motion_out akan mengikuti nilai dari sinyal car_detected. Dengan adanya nilai dari car_detected akan mengaktifkan keypad

- Password Keypad yang digunakan terdiri dari 3 digit, masing-masing dalam format heksadesimal dengan nilai antara 0 hingga F. Password untuk masuk adalah 123 (diwakili dalam biner sebagai 12'b000100100011), sementara password untuk keluar adalah 234 (diwakili dalam biner sebagai 12'b001000110100). Setiap digit dari password dimasukkan melalui sinyal biner digit1, digit2, dan digit3.
- Proses verifikasi password dilakukan secara sinkron dengan sinyal jam (clock). Ketika tombol lock_input ditekan, sistem akan memeriksa apakah kombinasi digit yang dimasukkan sesuai dengan salah satu dari password yang telah ditentukan (password masuk atau keluar). Jika cocok, sinyal unlock akan diaktifkan (1'b1), yang menunjukkan bahwa akses berhasil diberikan. Sebaliknya, jika kombinasi tidak cocok, atau jika tombol lock_input tidak ditekan, maka sinyal unlock akan tetap tidak aktif (1'b0).
- Dengan berhasilnya dimasukkan password keypad, maka akan membuka palang pintu (servo). Sistem menggunakan dua posisi utama untuk servo, SERVO_OPEN: Servo dalam keadaan terbuka, direpresentasikan dengan nilai logika 1'b1.
  SERVO_CLOSE: Servo dalam keadaan tertutup, direpresentasikan dengan nilai logika 1'b0
  Ketika sinyal reset aktif, nilai keluaran servo_pwm akan diatur ke posisi default, yaitu SERVO_CLOSE (tertutup). Namun, saat sinyal reset tidak aktif, sistem akan menentukan posisi servo berdasarkan sinyal control_signal. Jika control_signal aktif (logika tinggi), servo akan berpindah ke posisi SERVO_OPEN. Sebaliknya, jika control_signal tidak aktif (logika rendah), servo akan berada dalam posisi SERVO_CLOSE
- Dengan terbukanya palang pintu (servo) akan melakukan sistem counterpark yang digunkana untuk melakukan perhitungan nilai

berdasarkan sinyal a dan b, dengan kondisi untuk increment dan decrement. Dengan keluaran Menyimpan nilai penghitung ke dalam variabel occupancy. Dengan nilai penghitung diperbarui setiap siklus jam dan disalin ke variabel occupancy.

- Lalu dengan output dari counterpark akan memberikan masukan untuk parking_slot yang dimana akan memberikan indikasi jika parking_slot terisi atau kosong. parkir dengan menampilkan status slot menggunakan LED. Modul ini memanfaatkan nilai occupancy (jumlah kendaraan yang terparkir) untuk menentukan slot mana yang terisi, kemudian mengaktifkan LED yang sesuai pada output led_slot.

b. Pseudocode and Flowchart



➢ Start

➢ Motion sensor mendeteksi ada/tidaknya kendaraan

➢ Jika ada, maka akan mengaktifkan keypad

- ➢ Jika tidak ada, maka akan terus mendeteksi ada/tidaknya kendaraan
- ➢ Jika ada, kendaraan lalu PIN keypad berhasil di masukkan akan membuka palang pintu (Servo)
- ➢ Jika ada, kendaraan lalu PIN keypad salah dimasukkan akan tetap menutup palang pintu (Servo)
- ➢ Jika kendaraan berhasil membuka palang pintu (Servo), maka counter parking akan menambah slot_occupancy (slot parkir yang ditempati)
- ➢ Parking slot monitoring akan mendeteksi jika lahan parkir sudah terisi
- ➢ Jika Terisi akan memberikan output berupa sinyal 1 untuk menyalakan LED (yang menandakan slot sudah ditempati)
- ➢ Jika Tidak Terisi akan memberikan output berupa sinyal 0 untuk LED (yang menandakan slot kosong)
- ➢ Jika kendaraan ingin meninggalkan slot parkiran maka slot_occupancy akan dikurangi (menandakan berkurangnya slot yang terisi)

c. Pembagian beberapa modul

- Motion Sensor : Mendeteksi kendaraan yang masuk melaui sensor, jika terdeteksi kendaran dalam dalam radius sensor maka nilai dari motion sensor nya akan menjadi 1 dan jika tidak mendeteksi kendaraan apapun maka nilai dai motion sensor akan bernilai 0.
- Keypad : Input password setelah terdeteksi kendaraan di motion sensor, password pada keypad sendiri berbentuk 12 biner yang dibagi menjadi 3 digit angka, maka jika password salah unlock akan bernilai 0 dan jika password nya benar unlock akan bernilai 1.
- Keypad Handling : Submodul ini menggabungkan antara logic motion sensor dengan logic keypad, jika motion sensor bernilai 1 (kendaraan terdeteksi) maka input keypad akan bernilai 1, dan jika bernilai 0 maka input keypad akan bernilai 0, gunanya hal ini adalah untuk mengimplementasikan jika ada kendaraan di gate maka keypad akan

seolah olah muncul dan baru masuk ke dalam logika password yang berada di submodul keypad.

- Servo Control : Mengatur buka tutup nya gerbang pada sistem parkir.
- Servo Handler : Submodul ini mengatur logika dari keypad handling dan servo control untuk membuka dan menutup gate pada sistem parkir, jika password pada keypad benar maka servo pwm akan bernilai 1 (terbuka) jika password pada keypad salah maka servo pwm akan bernilai 0 (tertutup).
- Counter Parking: Menghitung slot kendaraan yang keluar dan masuk.
- Parking Slot Monitoring: Modul ini menerima input dari output Counter parking yang dimana berfungsi untuk mengatur slot parkir yang dilimit menjadi 20 slot parkiran saja, setiap slot parkir terdapat LED yang akan menyala jika terisi dan mati jika slot parkir kosong.
- Parking Controller : Berfungsi untuk menggabungkan modul Counter parking dengan parking slot monitoring, menerima data input dari gate masuk dan keluar lalu outputnya dimasukan pada modul counter parking yg hasil nya akan diolah modul Parking Slot Monitoring.

1.2. How the System Works

a. Arsitektur Sistem

Sistem parkir pintar ini menggunakan kamera deteksi gerak untuk mengaktifkan keypad saat kendaraan mendekati pintu masuk. Dengan kata sandi yang benar, servo motor membuka palang pintu secara otomatis. Setiap slot parkir dilengkapi sensor magnet dan LED: menyala untuk slot kosong, mati untuk terisi. Sistem ini menyediakan informasi real-time ketersediaan parkir, mempermudah pengguna dan menghemat waktu.

b. Input, Output, dan Internal Signals

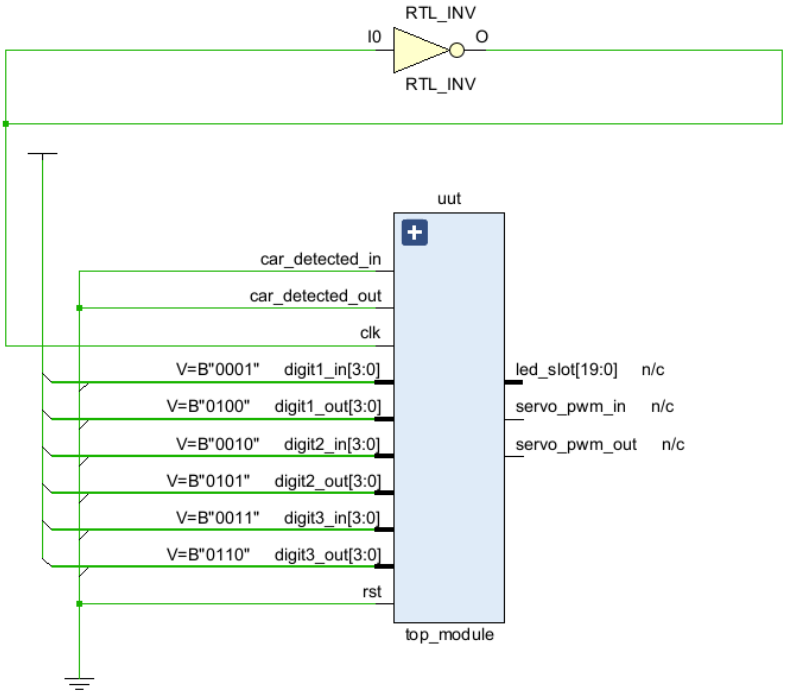
- Input : Car detected (motion sensor aktif), Keypad (input digit 1,digit 2,digit 3 menggunakan biner sebagai password), Lock Input (Mengaktifkan keypad setelah car detected), reset, servo pwm (input kendaraan masuk dan keluar untuk occupancy).

- Output : servo pwm, occupancy, unlock, led slot
- Internal Signal : password, slot parkir.

2. Design

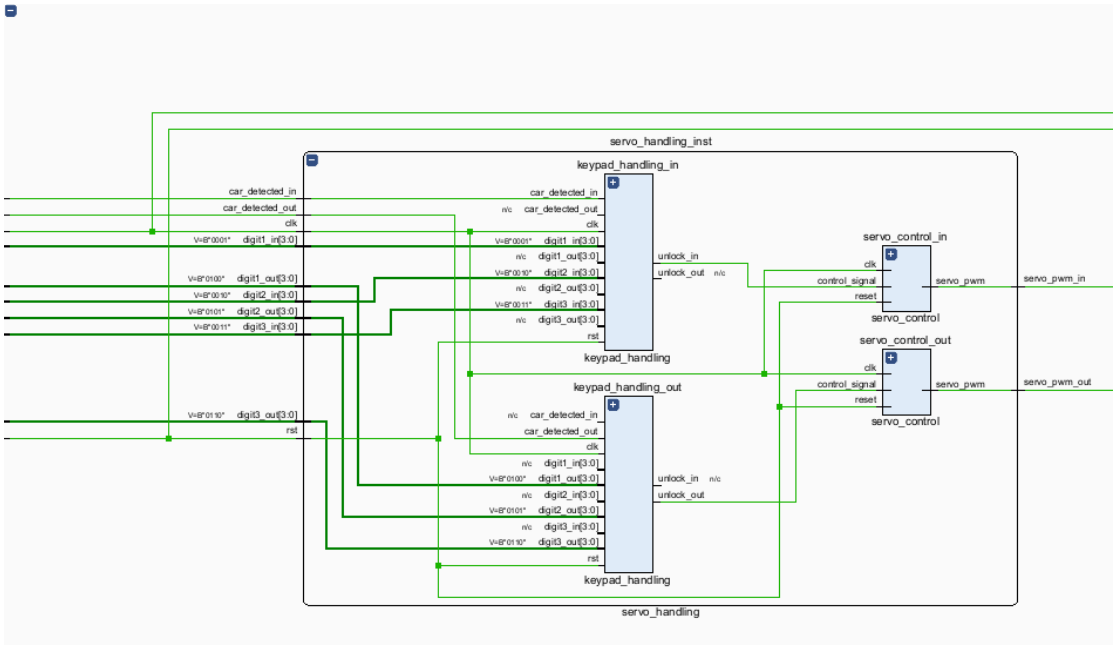
Bagian ini membahas tentang pembuatan dan penyusunan desain digital dari Smart Car Parking System

2.1. Block System 1



Blok 1 ini bertugas untuk menerima input berupa digit1_in[3:0], digit2_in[3:0], digit3_in[3:0] sebagai digit kode sandi (masing-masing 4 bit) dan car_detected_in sebagai sinyal deteksi keberadaan kendaraan. Blok ini memeriksa sinyal kehadiran kendaraan, di mana jika car_detected_in aktif (misalnya, bernilai 1), sistem akan melanjutkan proses pengkodean input. Setelah itu, input kode sandi (digit1, digit2, digit3) akan digabungkan dan diteruskan untuk divalidasi serta diterjemahkan menjadi data terkode yang siap digunakan oleh Processing Unit. Hasil pengkodean ini kemudian dikeluarkan melalui jalur keluaran ke blok pemrosesan berikutnya.

2.2. Block System 2



1. Sistem Utama

   Desain ini dibagi menjadi tiga blok utama:

   a. Keypad Handling untuk Masuk parkir (keypad_handling_in)

   b. Keypad Handling untuk Keluar dari parkir (keypad_handling_out)

   c. Servo Control (servo_control_in dan servo_control_out)

   Setiap blok saling terhubung untuk memproses sinyal dari perangkat input (keypad), mengontrol motor servo, dan mengelola operasi masuk serta keluar kendaraan pada sistem parkir.

2. Blok 1: Keypad Handling (Masuk dan Keluar)

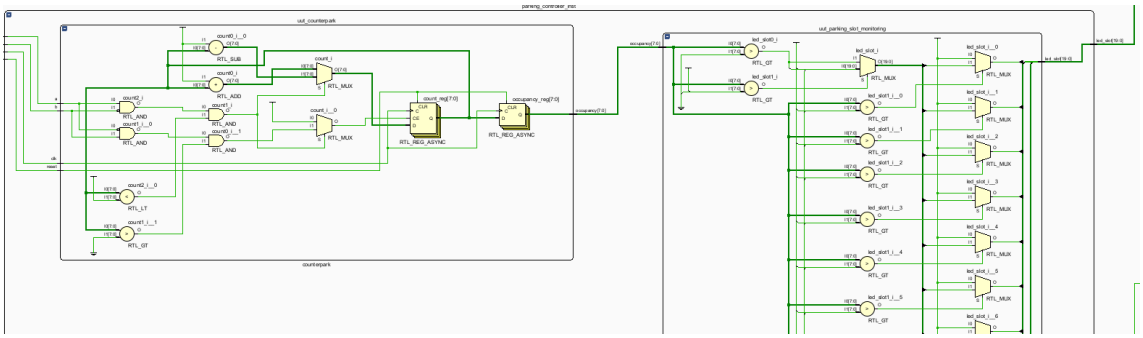
   a. Blok ini membandingkan password yang dimasukkan dengan password yang telah ditentukan untuk menentukan apakah akses diizinkan.

   b. input dari keypad (digit1, digit2, digit3) untuk memverifikasi password yang dimasukkan.

      Output: unlock_in atau unlock_out untuk memicu servo membuka/menutup gerbang.

3. Blok 2: Servo Control (Masuk dan Keluar)
    a. mengatur posisi motor servo (buka atau tutup) berdasarkan sinyal unlock dari keypad_handling.
    b. Output: servo_pwm_in dan servo_pwm_out untuk mengontrol motor servo pada gerbang masuk dan keluar.

2.3. Block System 3



1. Struktur Utama

    Sistem ini terdiri dari dua blok utama:
    a. Blok Counter: Menghitung jumlah kendaraan masuk dan keluar.
    b. Blok LED Display: Menampilkan status slot parkir berdasarkan jumlah kendaraan yang terdeteksi.

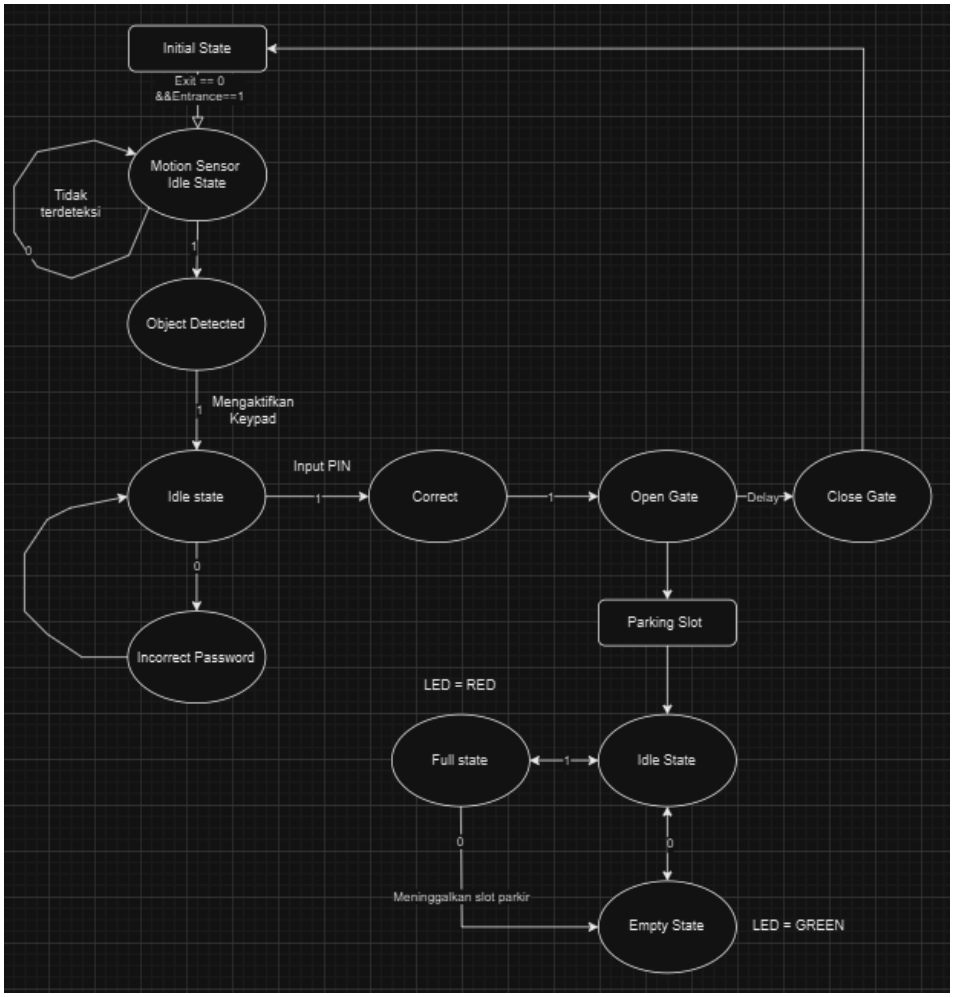
    Kedua blok saling terhubung untuk memastikan data jumlah kendaraan yang dihitung oleh Blok Counter dapat ditampilkan dengan benar pada Blok LED Display.

2. Blok 1: Counter
    a. Deteksi kendaraan dilakukan berdasarkan sinyal sensor (IN_A, IN_B, OUT_A, OUT_B) yang diproses untuk menghasilkan nilai OCCUPANCY (jumlah kendaraan saat ini di dalam parkiran).
    b. Output: Nilai occupancy (8-bit) yang menunjukkan jumlah kendaraan.

3. Blok 2: LED Display
    a. Menampilkan status slot parkir pada LED berdasarkan nilai output dari counter sebagai input occupancy yang dihasilkan dari Blok Counter. Setiap LED mewakili satu slot parkir. Jika slot terisi, LED menyala; jika kosong, LED mati.
    b. Output: Sinyal LED (LED_SLOT) untuk masing-masing slot parkir.

Blok Counter menghasilkan nilai occupancy berdasarkan jumlah kendaraan masuk dan keluar. Nilai occupancy ini diteruskan ke Blok LED Display untuk menyalakan atau mematikan LED sesuai status slot parkir.

2.4. FSM Design (Finite State Machine Design)



1. State and Transitions
    a. **Idle**: Sistem dalam keadaan menunggu input, baik dari sensor gerak atau keypad.
    b. **Keypad Idle**: Menunggu verifikasi PIN dari keypad.
    c. **Open Gate**: Membuka gerbang ketika verifikasi berhasil.
    d. **Parking Slot**: Menghitung kendaraan yang masuk atau keluar, memperbarui occupancy.
    e. **Close Gate**: Menutup gerbang setelah kendaraan masuk atau keluar.
    f. **Exit**: Kendaraan keluar dari parkiran, memperbarui occupancy

2. Verilog Code for FSM

```
module parking_fsm (
    input clk,
    input reset,
    input motion_detected,
    input pin_verified,
    input exit_signal,
    output reg open_gate,
    output reg close_gate,
    output reg [7:0] occupancy
);

    // State encoding
    typedef enum logic [2:0] {
        IDLE,
        KEYPAD_IDLE,
        OPEN_GATE,
        PARKING_SLOT,
        CLOSE_GATE,
        EXIT
    } state_t;

    state_t current_state, next_state;

    // Occupancy counter
    reg [7:0] slot_count;

    // State transition
    always @(posedge clk or posedge reset) begin
        if (reset)
            current_state <= IDLE;
        else
            current_state <= next_state;
    end

    // Next state logic
    always @(*) begin
        case (current_state)
```

```
            IDLE: begin
                if (motion_detected)
                    next_state = KEYPAD_IDLE;
                else
                    next_state = IDLE;
            end

            KEYPAD_IDLE: begin
                if (pin_verified)
                    next_state = OPEN_GATE;
                else
                    next_state = KEYPAD_IDLE;
            end

            OPEN_GATE: begin
                next_state = PARKING_SLOT;
            end

            PARKING_SLOT: begin
                next_state = CLOSE_GATE;
            end

            CLOSE_GATE: begin
                if (exit_signal)
                    next_state = EXIT;
                else
                    next_state = IDLE;
            end

            EXIT: begin
                next_state = IDLE;
            end

            default: next_state = IDLE;
        endcase
    end

    // Output logic
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            open_gate <= 0;
            close_gate <= 0;
            slot_count <= 0;
            occupancy <= 0;
        end else begin
            case (current_state)
                OPEN_GATE: begin
                    open_gate <= 1;
                    close_gate <= 0;
                end
                CLOSE_GATE: begin
                    open_gate <= 0;
                    close_gate <= 1;
```

```
                end
                PARKING_SLOT: begin
                    if (!exit_signal) begin
                        slot_count <= slot_count + 1;
                        occupancy <= slot_count;
                    end else begin
                        slot_count <= slot_count - 1;
                        occupancy <= slot_count;
                    end
                end
                default: begin
                    open_gate <= 0;
                    close_gate <= 0;
                end
            endcase
        end
    end

endmodule
```

3. Verilog Code For Top Module

    a. Top_Module

```
`timescale 1ns / 1ps
`include "servo_handling.v"
`include "parking_controller.v"

module top_module (
    input wire clk,
    input wire rst,
    input wire car_detected_in,       // Deteksi mobil masuk
    input wire car_detected_out,      // Deteksi mobil keluar
    input wire [3:0] digit1_in,       // Input digit pertama untuk jalur masuk
    input wire [3:0] digit2_in,       // Input digit kedua untuk jalur masuk
    input wire [3:0] digit3_in,       // Input digit ketiga untuk jalur masuk
    input wire [3:0] digit1_out,      // Input digit pertama untuk jalur keluar
    input wire [3:0] digit2_out,      // Input digit kedua untuk jalur keluar
    input wire [3:0] digit3_out,      // Input digit ketiga untuk jalur keluar
    output wire [19:0] led_slot,      // Output status LED slot parkir
    output wire servo_pwm_in,         // Output PWM untuk kontrol servo masuk
    output wire servo_pwm_out         // Output PWM untuk kontrol servo keluar
);

    wire unlock_in;
    wire unlock_out;

    servo_handling servo_handling_inst (
        .clk(clk),
        .rst(rst),
        .car_detected_in(car_detected_in),
```

```
        .car_detected_out(car_detected_out),
        .digit1_in(digit1_in),
        .digit2_in(digit2_in),
        .digit3_in(digit3_in),
        .digit1_out(digit1_out),
        .digit2_out(digit2_out),
        .digit3_out(digit3_out),
        .servo_pwm_in(servo_pwm_in),
        .servo_pwm_out(servo_pwm_out)
    );

    parking_controller parking_controller_inst (
        .clk(clk),
        .reset(rst),
        .a(servo_pwm_in),
        .b(servo_pwm_out),
        .led_slot(led_slot)
    );

endmodule
```

b. Testbench

```
`timescale 1ns / 1ps
`include "top_module.v"

module top_module_tb;

    reg clk;
    reg rst;
    reg car_detected_in;
    reg car_detected_out;
    reg [3:0] digit1_in;
    reg [3:0] digit2_in;
    reg [3:0] digit3_in;
    reg [3:0] digit1_out;
    reg [3:0] digit2_out;
    reg [3:0] digit3_out;
    wire [19:0] led_slot;
    wire servo_pwm_in;
    wire servo_pwm_out;

    top_module uut (
        .clk(clk),
        .rst(rst),
        .car_detected_in(car_detected_in),
        .car_detected_out(car_detected_out),
        .digit1_in(digit1_in),
        .digit2_in(digit2_in),
        .digit3_in(digit3_in),
```

```
        .digit1_out(digit1_out),
        .digit2_out(digit2_out),
        .digit3_out(digit3_out),
        .led_slot(led_slot),
        .servo_pwm_in(servo_pwm_in),
        .servo_pwm_out(servo_pwm_out)
    );

    always begin
        #5 clk = ~clk;
    end

    initial begin
        clk = 0;
        rst = 0;
        car_detected_in = 0;
        car_detected_out = 0;
        digit1_in = 4'b0000;
        digit2_in = 4'b0000;
        digit3_in = 4'b0000;
        digit1_out = 4'b0000;
        digit2_out = 4'b0000;
        digit3_out = 4'b0000;

        rst = 1;
        #50;
        rst = 0;

        // State 1: Mobil masuk (deteksi mobil, set input digit)
        car_detected_in = 1;
        digit1_in = 4'b0001;
        digit2_in = 4'b0010;
        digit3_in = 4'b0011;
        #100;

        // State 2: Mobil keluar (deteksi mobil keluar, set digit output)
        car_detected_in = 0;
        car_detected_out = 1;
        digit1_out = 4'b0010;
        digit2_out = 4'b0011;
        digit3_out = 4'b0100;
        #100;

        // State 3: Tidak ada mobil
        car_detected_out = 0;
        #100;

        // State 4: Mobil masuk dengan input digit yang berbeda
        car_detected_in = 1;
        digit1_in = 4'b1001;
        digit2_in = 4'b1010;
        digit3_in = 4'b1011;
        #100;
```

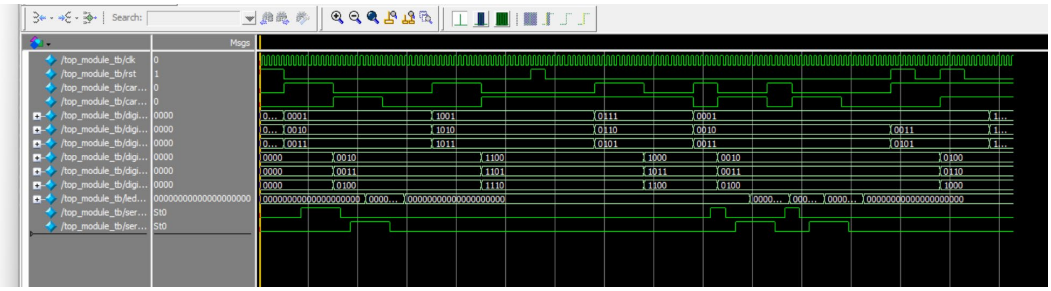
```
// State 5: Mobil keluar dengan input digit yang berbeda
car_detected_in = 0;
car_detected_out = 1;
digit1_out = 4'b1100;
digit2_out = 4'b1101;
digit3_out = 4'b1110;
#100;

// State 6: Reset sistem, cek jika LED slot dalam keadaan off
rst = 1;
#30;
rst = 0;
#100;

// State 7: Mobil masuk, cek led_slot
car_detected_in = 1;
digit1_in = 4'b0111;
digit2_in = 4'b0110;
digit3_in = 4'b0101;
#100;

// State 8: Mobil keluar setelah masuk, pastikan led_slot berubah
car_detected_in = 0;
car_detected_out = 1;
digit1_out = 4'b1000;
digit2_out = 4'b1011;
digit3_out = 4'b1100;
#100;

// State 9: Mobil masuk dan keluar, simulasi pergerakan berulang
car_detected_in = 1;
car_detected_out = 0;
digit1_in = 4'b0001;
digit2_in = 4'b0010;
digit3_in = 4'b0011;
#50;

car_detected_in = 0;
car_detected_out = 1;
digit1_out = 4'b0010;
digit2_out = 4'b0011;
digit3_out = 4'b0100;
#100;

// State 10: Mobil kembali masuk setelah keluar
car_detected_in = 1;
car_detected_out = 0;
digit1_in = 4'b0001;
digit2_in = 4'b0010;
digit3_in = 4'b0011;
#50;
```

```
        // State 11: Mobil keluar setelah beberapa waktu
        car_detected_in = 0;
        car_detected_out = 1;
        digit1_out = 4'b0010;
        digit2_out = 4'b0011;
        digit3_out = 4'b0100;
        #100;

        // State 12: Tidak ada mobil di jalur keluar
        car_detected_out = 0;
        #100;

        // State 13: Mobil masuk dengan kondisi reset, cek output LED slot
        rst = 1;
        car_detected_in = 1;
        digit1_in = 4'b0001;
        digit2_in = 4'b0011;
        digit3_in = 4'b0101;
        #50;
        rst = 0;
        #50;

        // State 14: Mobil keluar dengan kondisi reset, cek LED slot
        rst = 1;
        car_detected_out = 1;
        digit1_out = 4'b0100;
        digit2_out = 4'b0110;
        digit3_out = 4'b1000;
        #50;
        rst = 0;
        #50;

        // State 15: Semua input dalam kondisi normal setelah reset
        car_detected_in = 1;
        car_detected_out = 1;
        digit1_in = 4'b1010;
        digit2_in = 4'b1100;
        digit3_in = 4'b1110;
        #50;

    end
endmodule
```

3. Hardware Implementation

Bagian ini berisi transisi mulai dari design sampai testing dan implementasi pada hardware

3.1. Simulation (Waveform Analysis)



Sinyal Input :

- clk -> Sinyal clock berfungsi untuk sinkronisasi operasi rangkaian digital dan memicu tindakan tertentu di dalam sistem.
- rst -> Sinyal reset (rst) digunakan untuk mengembalikan semua register, flip-flop, atau state ke nilai awal saat sistem mulai bekerja.
- car_detected -> Sinyal ini berfungsi sebagai penanda bahwa ada kendaraan yang terdeteksi di depan palang parkir (masuk/keluar).
- digit(1, 2 3) -> Sinyal ini berfungsi sebagai input password (masuk/keluar) dari keypad yang ditekan.

Sinyal Output :

- led_slot -> Sinyal ini berfungsi sebagai output yang mengindikasikan slot parkir ke berapa yang telah terisi.
- servo_pwm -> Sinyal ini adalah output yang berfungsi sebagai input dari counter parking, yang mana jika servo_pwm_in yang aktif maka led_slot pada bit tertentu akan bernilai 1 dan jika servo_pwm_out yang aktif maka sebaliknya.

3.2. Implementation Level 1 (Easy)

- **Motion sensor module :** mengimplementasikan modul motion sensor dengan output motion_out yang akan menjadi input untuk terbukanya keypad.
- **Keypad module :** mengimplementasikan modul keypad dengan 3 digit input dan memiliki output unlock sebagai input dari servo yang memberi perintah untuk terbukanya servo.
- **Servo module :** mengimplementasikan modul servo dengan input yang dikirim dari keypad modul sekaligus mengatur terbuka dan tertutupnya servo.
- **Parking slot monitoring :** mengimplementasikan modul parking slot sebagai indikasi perubahan status pada slot parkir sekaligus output terakhir dari sistem.

3.3. Implementation Level 2 (Moderate)

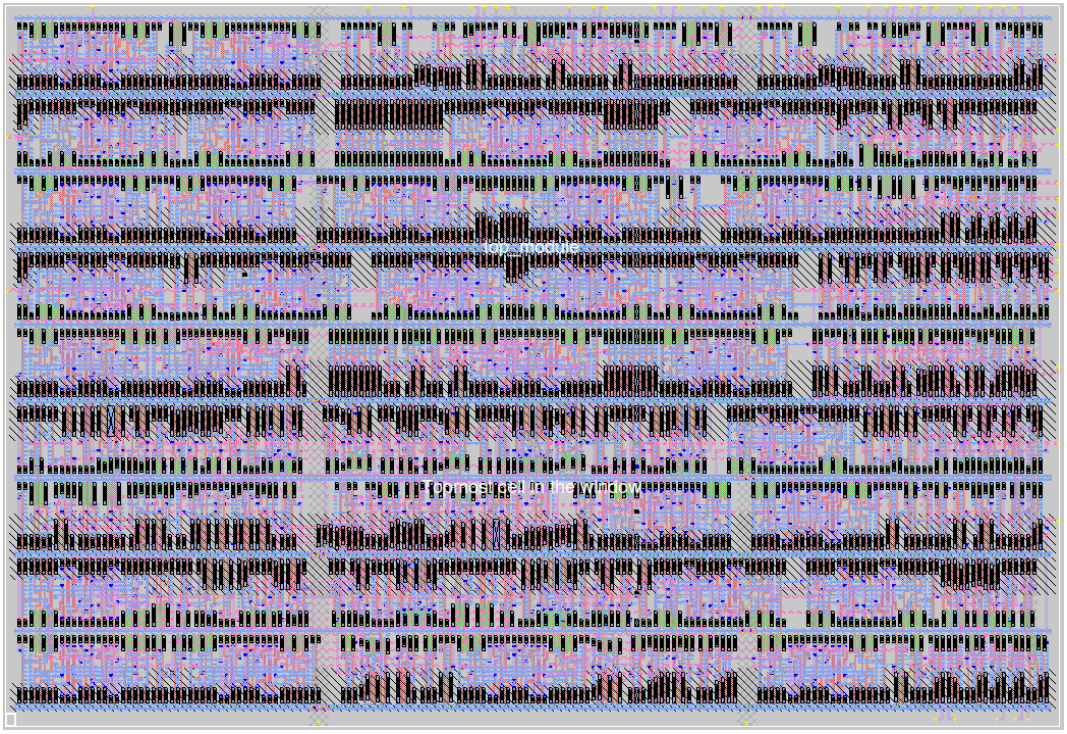
- **Keypad Handling :** Mengintegrasikan modul keypad dan motion sensor dalam satu file.
- **Servo Handling :** Mengintegrasikan modul keypad handling dan servo dalam satu file.
- **Parking Controller :** Mengintegrasikan modul counter parking dan slot parking dalam satu file.

3.4. Implementasi Level 3

- **Top Module :** Mengintegrasikan semua modul yang telah ada di servo handling dan parking controller dalam satu file.

3.5. Qflow Implementation (ASIC Semi-Custom)

3.5.1. Place-and-route



Proses routing untuk desain top_module di Qflow berhasil diselesaikan tanpa kegagalan jalur, dengan total 1561 jalur yang berhasil dirouting. Data LEF dan DEF dibaca dengan baik. Jalur penting seperti vdd, rst, dan sinyal dari parking_controller_inst berhasil dirouting, termasuk sinyal jam (clk) dan berbagai input/output lainnya. File DEF yang dihasilkan mencakup 284 jalur dan 3 jalur khusus.

3.5.2. power constraints for a semi-custom ASIC design.

Total On-Chip Power:	**0.068 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.3°C**
Thermal Margin:	59.7°C (11.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	High

1. Total Daya On-Chip
   - Total On-Chip Power: 0.068 W
     - Desain ini mengkonsumsi daya yang relatif rendah, yang menunjukkan efisiensi daya yang baik. Ini penting untuk aplikasi yang memerlukan penghematan energi.

2. Suhu Junction
   - Junction Temperature: 25.3°C
     - Suhu junction yang rendah menunjukkan bahwa perangkat beroperasi dalam kondisi yang aman dan tidak mengalami overheating. Ini penting untuk menjaga kinerja dan umur panjang perangkat.
3. Margin Termal
   - Thermal Margin: 59.7°C (11.9 W)
     - Margin termal yang tinggi menunjukkan bahwa desain memiliki kapasitas untuk menangani lebih banyak daya sebelum mencapai batas suhu kritis. Ini memberikan fleksibilitas dalam pengembangan dan potensi untuk meningkatkan kinerja jika diperlukan.
4. Suhu Ambient
   - Ambient Temperature: 25.0°C

- Suhu lingkungan yang normal menunjukkan bahwa perangkat beroperasi dalam kondisi yang ideal, yang berkontribusi pada kinerja yang stabil.

5. Efektif SjA
   - Effective SjA: 5.0°C/W
     - Nilai ini menunjukkan seberapa banyak suhu junction akan meningkat per watt daya yang dikonsumsi. Nilai yang lebih rendah menunjukkan desain yang lebih efisien dalam mengelola panas.
6. Tingkat Kepercayaan
   - Confidence level: High
     - Tingkat kepercayaan yang tinggi menunjukkan bahwa analisis daya ini dapat diandalkan dan memberikan gambaran akurat tentang konsumsi daya desain.

Desain pada board Basys3 menunjukkan efisiensi daya yang baik dengan total konsumsi daya yang rendah dan suhu junction yang aman. Margin termal yang tinggi memberikan fleksibilitas untuk pengembangan lebih lanjut. Meskipun tidak ada anggaran daya yang ditentukan, penting untuk terus memantau konsumsi daya agar tetap dalam batas yang dapat diterima untuk aplikasi yang lebih besar atau lebih kompleks.

3.5.3 Sel (Cells) yang Digunakan

a. Buffer Cells:
   - BUFX4: Buffer dengan penguatan 4x, digunakan untuk meningkatkan sinyal.
   - BUFX2: Buffer dengan penguatan 2x, juga digunakan untuk meningkatkan sinyal.
   - CLKBUF1: Buffer khusus untuk sinyal clock.

b. Inverter Cells:
   - INVX1: Inverter (NOT gate) dengan delay 100p.
   - INVX2: Inverter dengan delay 100p.
   - INVX4: Inverter dengan delay 100p.
   - INVX8: Inverter dengan delay 100p.

c. Logic Gates:
    - NOR2X1: 2-input NOR gate.
    - NAND2X1: 2-input NAND gate.
    - NAND3X1: 3-input NAND gate.
    - AND2X2: 2-input AND gate.
    - AOI21X1: AND-OR-Invert gate (2-input AND, 1-input OR).
    - OAI21X1: OR-AND-Invert gate (2-input OR, 1-input AND).
    - XOR2X1: 2-input XOR gate.
    - XNOR2X1: 2-input XNOR gate.
    - OR2X2: 2-input OR gate.
d. Multiplexer:
    - MUX2X1: 2-input multiplexer.
    - D Flip-Flops:
    - DFFSR: D flip-flop dengan reset asinkron.
e. Analog Models:
    - todig_1v8: Model ADC untuk konversi analog ke digital.
    - toana_1v8: Model DAC untuk konversi digital ke analog.
    -

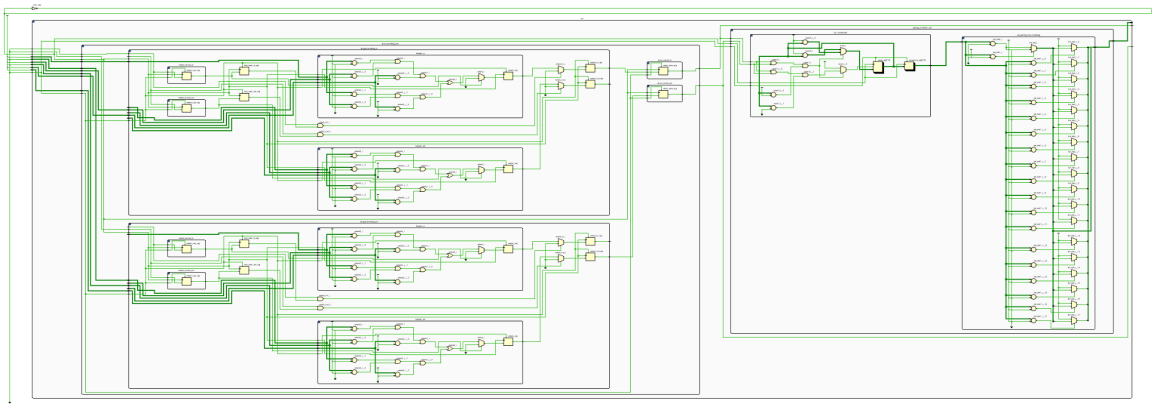
Netlist ini menggambarkan desain sistem kontrol parkir yang melibatkan pengolahan sinyal digital untuk mendeteksi mobil masuk dan keluar, serta mengontrol LED dan servo. Buffer digunakan untuk memperkuat sinyal clock dan sinyal lainnya agar tidak terjadi penurunan tegangan saat sinyal melewati rangkaian. Berbagai gerbang logika (NAND, NOR, AND, OR) digunakan untuk mengimplementasikan logika kontrol yang diperlukan untuk sistem parkir. D flip-flops digunakan untuk menyimpan status dari sinyal yang berhubungan dengan deteksi mobil dan status kunci. Selain itu, model ADC dan DAC digunakan untuk menghubungkan sinyal digital dengan sinyal analog, memungkinkan interaksi dengan perangkat keras lainnya.

4. Results

Bagian ini digunakna untuk dokumentasi dan analisis hasil

4.1. Network Architecture

4.1.1. RTL Design



1. Modul Utama
    a. Deteksi Input :
        - Modul : Motion Sensor
        - Fungsi :
            1. Motion sensor digunakan untuk mendeteksi keberadaan kendaraan pada jalur masuk dan menghasilkan sinyal keluaran bernama motion_out. Sinyal motion_out menjadi penghubung antara motion sensor dan keypad, memastikan bahwa keypad hanya aktif jika kendaraan terdeteksi di jalur masuk.

    b. Password handling :
        - Modul : Keypad dan Keypad Handling
        - Fungsi :
            1. untuk memasukkan password yang diperlukan setelah kendaraan terdeteksi oleh motion sensor. Password yang dimasukkan melalui Keypad berbentuk 12-bit biner yang kemudian dibagi menjadi 3 digit angka desimal.

a. Sistem akan memeriksa validitas password tersebut dengan mencocokkannya terhadap password yang benar.

2. mengintegrasikan logika motion sensor dengan logika pada Keypad. Tujuannya adalah memastikan bahwa Keypad hanya aktif dan dapat menerima input jika kendaraan terdeteksi di area gate oleh motion sensor.

c. Control Output
   - Modul : CounterPark
   - Fungsi :
     1. Mendukung deteksi kendaraan masuk dan keluar dengan akurasi tinggi.
     2. Mengelola nilai occupancy secara otomatis berdasarkan sinyal dari sensor.
     3. Memastikan pembaruan data penghitung pada setiap siklus clock, menjaga data selalu terkini.

2. Interaksi Input dan Output :
   a. Dari modul servo_handling :
      - Modul ini menerima input car_detected_in, car_detected_out, serta kode digit (digit1_in, digit2_in, digit3_in, digit1_out, digit2_out, digit3_out).
      - Berdasarkan input tersebut, modul ini menghasilkan sinyal servo_pwm_in dan servo_pwm_out untuk menggerakkan servo pintu masuk dan keluar.

   b. Dari modul parking_controller :
      - Modul ini menerima sinyal servo_pwm_in dan servo_pwm_out dari modul servo_handling.
      - Berdasarkan sinyal tersebut, modul ini mengelola status slot parkir dan memperbarui output led_slot, yang menunjukkan ketersediaan parkir.

4.2. Optimization, Complexity, and Other Observable Variables

4.2.1. Analisis Efisiensi Design

1. Timing

Analisis waktu statik dilakukan menggunakan alat Vesta untuk mengevaluasi kinerja desain top_module. Berikut adalah hasil utama dari analisis ini:

1. Jumlah Jalur yang Dianalisis: 66 jalur
2. Frekuensi Jam Maksimum yang Dihitung: 323.66 MHz

a. Jalur dengan Delay Maksimum

Berikut adalah 20 jalur dengan penundaan maksimum:

- Jalur DFFSR_6/CLK ke DFFSR_2/D:
  - Total Penundaan: 3089.66 ps
  - Skew Jam: 3.28 ps
  - Setup di Tujuan: 201.158 ps
- Jalur DFFSR_6/CLK ke DFFSR_3/D:
  - Total Penundaan: 3089.66 ps
  - Skew Jam: 3.28 ps
  - Setup di Tujuan: 201.158 ps
- Jalur DFFSR_6/CLK ke DFFSR_4/D:
  - Total Penundaan: 3089.66 ps
  - Skew Jam: 3.28 ps
  - Setup di Tujuan: 201.158 ps
- Jalur DFFSR_6/CLK ke DFFSR_5/D:
  - Total Penundaan: 3089.66 ps
  - Skew Jam: 3.28 ps
  - Setup di Tujuan: 201.158 ps
- Jalur DFFSR_6/CLK ke DFFSR_7/D:
  - Total Penundaan: 3089.66 ps
  - Skew Jam: 3.28 ps
  - Setup di Tujuan: 201.158 ps

b. Jalur dengan Penundaan Minimum

Berikut adalah 20 jalur dengan penundaan minimum:

- Jalur input pin digit3_out[1] ke DFFSR_27/D:
    - Total Penundaan: 678.511 ps
    - Setup di Tujuan: 177.062 ps
- Jalur input pin digit3_in[1] ke DFFSR_23/D:
    - Total Penundaan: 678.511 ps
    - Setup di Tujuan: 177.062 ps
- Jalur input pin clk ke DFFSR_18/CLK:
    - Total Penundaan: 565.875 ps
    - Setup di Tujuan: 353.96 ps

Desain top_module memenuhi persyaratan waktu minimum untuk hold dan setup. Namun, jalur dengan penundaan maksimum menunjukkan bahwa desain beroperasi pada frekuensi maksimum 323.66 MHz, yang perlu diperhatikan untuk memastikan kinerja yang optimal dalam aplikasi nyata.

2. Penggunaan Resource
    a. Logika Element :
        - FSM: Terdiri dari enam state (IDLE, KEYPAD_IDLE, OPEN_GATE, PARKING_SLOT, CLOSE_GATE, EXIT), membutuhkan jumlah gerbang logika yang minimal.
        - Keypad Handling: Menggunakan logika kombinasi untuk verifikasi password, yang ringan dalam penggunaan sumber daya.
        - Counter: Menggunakan register 8-bit untuk menghitung okupansi.
        - LED Display: Menggunakan register 20-bit untuk melacak status slot parkir, serta logika decoding untuk mengontrol LED.
    b. Memory:

        Desain ini tidak menggunakan banyak sumber daya memori. Penanganan password hanya memerlukan beberapa register untuk penyimpanan sementara input 12-bit dan logika perbandingan.

3. Konsumsi Daya

   FPGA dengan desain sederhana seperti ini (misalnya ModelSim atau Intel Cyclone IV) memiliki konsumsi daya dalam rentang:

   - Statik (Idle): 50–200 mW
   - Dinamis (Aktif): 300–800 mW (tergantung jumlah elemen logika yang digunakan dan aktivitas switching).

   b. **Kontributor Utama**:

   - FPGA
     - Idle Mode: 100–150 mW.
     - Active Mode: 400–600 mW.
   - Motion Sensor ~20–50 mW
   - Servo (melalui driver)
   - Keypad ~10–20 mW
   - LED untuk Slot Parkir: ~20 mW per LED

   Total konsumsi daya :
   Idle Mode (tanpa motor aktif):
   ~600–800 mW (tergantung penggunaan LED dan aktivitas FPGA).
   Active Mode (dengan motor aktif):
   ~2–3 W (termasuk konsumsi daya motor servo).

4. *Trade-offs*

   a. Performa vs. Area:

   - Performa tinggi diprioritaskan untuk respons sistem waktu nyata.
   - Penggunaan area sedikit lebih besar karena adanya FSM, counter, dan logika kontrol tambahan untuk servo handling. Namun, desain ini tetap kompak untuk implementasi FPGA/ASIC.

5. Batasan :

   - Skalabilitas: Sistem saat ini mendukung hingga 20 slot parkir. Untuk parkiran yang lebih besar, dibutuhkan sumber daya tambahan untuk pengelolaan memori dan tampilan.

5. Conclusion

Setelah berhasil menggabungkan semua modul dalam sistem smart parking, kami dapat menyimpulkan bahwa logika sistem bekerja sesuai dengan teori dasar yang telah dirancang. Hasil simulasi menunjukkan bahwa sistem mampu merespons perubahan dengan baik, mencerminkan ketepatan implementasi logika yang dirancang. Dengan demikian, sistem ini membuktikan kemampuan untuk mengelola dan memonitor kondisi parkir secara efisien sesuai dengan spesifikasi yang diharapkan.

6. Full Documentation, Source Code, and Images

https://github.com/dn3ndra/smart-parking-system-verilog.git