Adaptive Traffic Lights

Kelompok AP05

Anggota Kelompok:

1. Christopher Sutandar (2206810414)
2. Kevin Ariono (2206059603)
3. Phoebe Ivana (2206820320)
4. Surya Dharmasaputra Soeroso (2206827825)

Latar Belakang

Pertumbuhan jumlah kendaraan dan mobilitas penduduk di perkotaan telah menyebabkan tantangan besar dalam pengelolaan lalu lintas. Perempatan jalan menjadi titik fokus yang memerlukan sistem pengaturan yang efektif untuk menjaga kelancaran arus kendaraan, keselamatan pengguna jalan, dan penggunaan ruang jalan yang optimal. Berdasarkan permasalahan tersebut, pengembangan sistem pengatur lalu lintas pada perempatan menjadi esensial. Dengan mengintegrasikan konsep – konsep sistem digital pada pemrograman berbahasa VHDL, harapannya sistem mampu memberikan solusi yang efisien dan adaptif dalam mengatur lalu lintas.

Deskripsi Proyek

Pengatur Lalu Lintas Perempatan adalah sebuah device yang dibuat berdasarkan bahasa pemrograman VHDL. Fungsi yang terdapat pada lampu lalu lintas ini ada *display countdown*, sistem penyeberangan khusus untuk para lansia, dan sistem adaptasi terhadap kepadatan lalu lintas. Kemampuannya untuk mengukur kepadatan lalu lintas dan menyesuaikan durasi lampu secara otomatis memainkan peran penting dalam mengoptimalkan aliran kendaraan

Tujuan

1. Membuat sistem lampu lalu lintas yang dapat beradaptasi terhadap tingkat kepadatan jalur serta pengguna jalan
2. Menambahkan fitur penyeberangan untuk lansia
3. Menambahkan fitur kepadatan jalan
4. Mengaplikasikan semua modul ke dalam program

Alat yang Digunakan

1. VS Code
2. ModelSim
3. Intel Quartus Prime

Implementasi Program

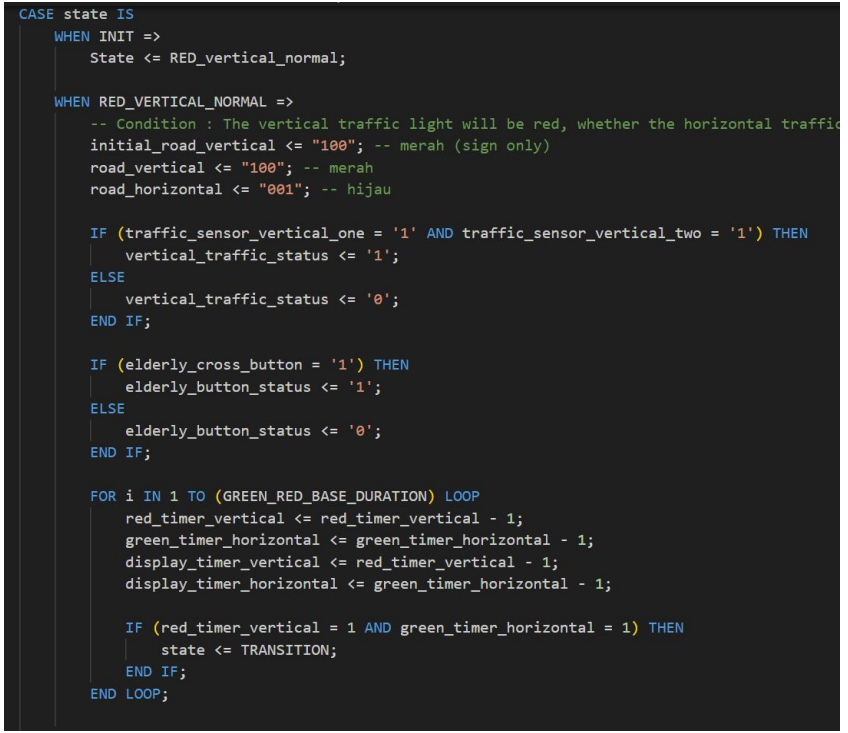
Decoder

Entitas decoder bertugas mengkonversi bilangan desimal menjadi representasi seven segment yang diperlukan untuk menampilkan digit pada layar LED. Hal ini dilakukan dengan memetakan setiap digit dalam rentang 0 hingga 50 menjadi representasi seven segment yang sesuai.

```
PROCESS (digit1, digit2)
BEGIN
    CASE digit1 IS
        WHEN 0 => seg1_out <= "0000001"; -- Corresponding segments for digit 0
        WHEN 1 => seg1_out <= "1001111"; -- Corresponding segments for digit 1
        WHEN 2 => seg1_out <= "0010010"; -- Corresponding segments for digit 2
        WHEN 3 => seg1_out <= "0000110"; -- Corresponding segments for digit 3
        WHEN 4 => seg1_out <= "1001100"; -- Corresponding segments for digit 4
        WHEN 5 => seg1_out <= "0100100"; -- Corresponding segments for digit 5
        WHEN 6 => seg1_out <= "0100000"; -- Corresponding segments for digit 6
        WHEN 7 => seg1_out <= "0001111"; -- Corresponding segments for digit 7
        WHEN 8 => seg1_out <= "0000000"; -- Corresponding segments for digit 8
        WHEN 9 => seg1_out <= "0000100"; -- Corresponding segments for digit 9
        WHEN OTHERS => seg1_out <= "1111111"; -- All segments off for other values
    END CASE;

    CASE digit2 IS
        WHEN 0 => seg2_out <= "0000001"; -- Corresponding segments for digit 0
        WHEN 1 => seg2_out <= "1001111"; -- Corresponding segments for digit 1
        WHEN 2 => seg2_out <= "0010010"; -- Corresponding segments for digit 2
        WHEN 3 => seg2_out <= "0000110"; -- Corresponding segments for digit 3
        WHEN 4 => seg2_out <= "1001100"; -- Corresponding segments for digit 4
        WHEN 5 => seg2_out <= "0100100"; -- Corresponding segments for digit 5
        WHEN 6 => seg2_out <= "0100000"; -- Corresponding segments for digit 6
        WHEN 7 => seg2_out <= "0001111"; -- Corresponding segments for digit 7
        WHEN 8 => seg2_out <= "0000000"; -- Corresponding segments for digit 8
        WHEN 9 => seg2_out <= "0000100"; -- Corresponding segments for digit 9
        WHEN OTHERS => seg2_out <= "1111111"; -- All segments off for other values
    END CASE;
```

State Ketika Lampu Vertikal Merah

Entitas Adaptive Traffic Light lebih kompleks dan berfungsi untuk mengatur logika lampu lalu lintas yang adaptif. Ini melibatkan variabel dan sinyal yang merepresentasikan waktu, keadaan jalan, dan status sensor seperti kepadatan lalu lintas atau tombol untuk lansia. Dengan menggunakan logika state machine, entitas ini mengatur perilaku lampu lalu lintas berdasarkan kondisi jalan yang dideteksi. Misalnya, ketika kondisi tertentu terpenuhi, lampu akan berubah dari merah ke hijau atau sebaliknya, sesuai dengan kondisi yang diatur.



```
CASE state IS
    WHEN INIT =>
        State <= RED_vertical_normal;

    WHEN RED_VERTICAL_NORMAL =>
        -- Condition : The vertical traffic light will be red, whether the horizontal traffic
        initial_road_vertical <= "100"; -- merah (sign only)
        road_vertical <= "100"; -- merah
        road_horizontal <= "001"; -- hijau

        IF (traffic_sensor_vertical_one = '1' AND traffic_sensor_vertical_two = '1') THEN
            vertical_traffic_status <= '1';
        ELSE
            vertical_traffic_status <= '0';
        END IF;

        IF (elderly_cross_button = '1') THEN
            elderly_button_status <= '1';
        ELSE
            elderly_button_status <= '0';
        END IF;

        FOR i IN 1 TO (GREEN_RED_BASE_DURATION) LOOP
            red_timer_vertical <= red_timer_vertical - 1;
            green_timer_horizontal <= green_timer_horizontal - 1;
            display_timer_vertical <= red_timer_vertical - 1;
            display_timer_horizontal <= green_timer_horizontal - 1;

            IF (red_timer_vertical = 1 AND green_timer_horizontal = 1) THEN
                state <= TRANSITION;
            END IF;
        END LOOP;
```

Signal dan konstanta yang digunakan

Variabel dan konstanta yang tertera digunakan untuk menentukan durasi dari setiap fase lampu lalu lintas berdasarkan kondisi tertentu seperti lansia menekan tombol lintas, kepadatan lalu lintas yang tinggi, atau kondisi awal sistem. Dengan mengatur durasi dari fase-fase lampu lalu lintas, logika ini dapat menyesuaikan perilaku lampu lalu lintas secara adaptif sesuai dengan kondisi jalan yang dideteksi.

```
SIGNAL green_timer_vertical, red_timer_horizontal : INTEGER := 10;
SIGNAL red_timer_vertical, green_timer_horizontal : INTEGER := 10;
SIGNAL yellow_timer_vertical, yellow_timer_horizontal : INTEGER := 5;
SIGNAL initial_road_vertical : STD_LOGIC_VECTOR(2 DOWNTO 0) := "100";
SIGNAL display_timer_vertical, display_timer_horizontal : INTEGER RANGE 0 TO 59;

SIGNAL elderly_button_status, vertical_traffic_status, horizontal_traffic_status : STD_LOGIC;

CONSTANT YELLOW_BASE_DURATION : INTEGER := 5;
CONSTANT GREEN_RED_BASE_DURATION : INTEGER := 10;
CONSTANT GREEN_RED_BASE_DURATION_ELDERLY : INTEGER := 15;
CONSTANT GREEN_RED_BASE_DURATION_TRAFFIC : INTEGER := 15;
```

Algoritma fitur

Bagian ini dalam kode mengatur durasi fase-fase lampu lalu lintas secara adaptif. Jika tombol menyebrang lansia ditekan, durasi fase merah-hijau untuk lansia diterapkan, memberi lebih banyak waktu bagi mereka untuk menyeberang. Selain itu, jika terdeteksi kepadatan lalu lintas di jalur vertikal atau horizontal, durasi fase hijau diperpanjang untuk jalur yang padat, mengoptimalkan waktu bagi setiap jenis lalu lintas untuk bergerak dengan lebih efisien. Dengan responsif terhadap kondisi lalu lintas aktual, sistem memastikan keselamatan yang lebih baik dan aliran lalu lintas yang lebih lancar di persimpangan jalan.

```
IF (elderly_cross_button = '1') THEN
    red_timer_vertical <= GREEN_RED_BASE_DURATION_ELDERLY;
    green_timer_horizontal <= GREEN_RED_BASE_DURATION_ELDERLY;
    green_timer_vertical <= GREEN_RED_BASE_DURATION_ELDERLY;
    red_timer_horizontal <= GREEN_RED_BASE_DURATION_ELDERLY;
END IF;

IF (vertical_traffic_status = '1') THEN
    green_timer_vertical <= GREEN_RED_BASE_DURATION_TRAFFIC;
    red_timer_horizontal <= GREEN_RED_BASE_DURATION_TRAFFIC;
END IF;

IF (horizontal_traffic_status = '1') THEN
    green_timer_horizontal <= GREEN_RED_BASE_DURATION_TRAFFIC;
    red_timer_vertical <= GREEN_RED_BASE_DURATION_TRAFFIC;
END IF;
```

Kutipan Kode Testbench

Komponen testbench bertujuan untuk menguji simulasi perilaku dari entitas Adaptive Traffic Light. Testbench menyediakan sinyal clock (clk) dan sinyal input acak yang mencoba mensimulasikan berbagai situasi di jalan seperti sensor lalu lintas yang berubah-ubah atau tombol lansia yang ditekan. Dengan memberikan sinyal-sinyal ini, testbench dapat mengamati dan merekam bagaimana entitas Adaptive Traffic Light bereaksi terhadap perubahan situasi yang disimulasikan.

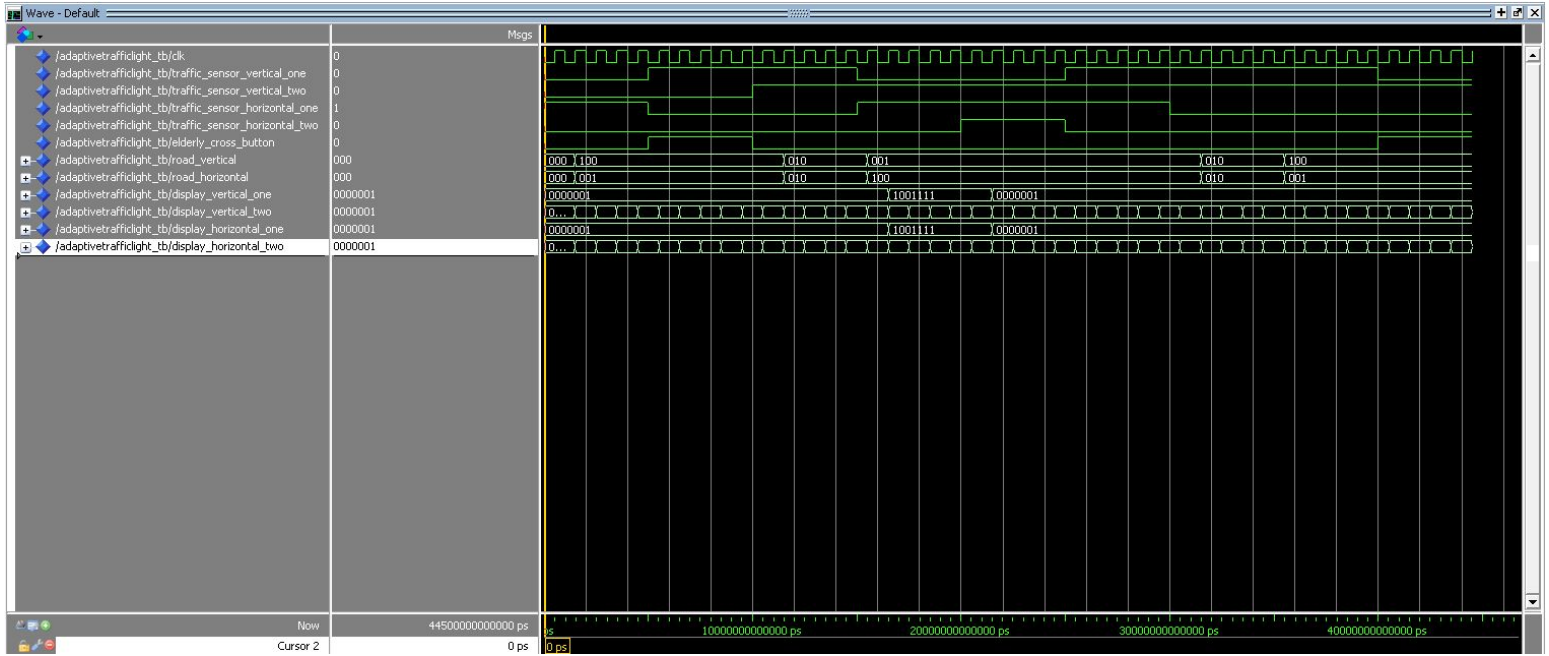
```
-- Clock process to simulate real-time behavior
CLK_PROCESS : PROCESS
BEGIN
    WHILE (TRUE) LOOP
        clk <= '0';
        WAIT FOR 500 ms; -- Adjust this value to simulate real-time behavior
        clk <= '1';
        WAIT FOR 500 ms; -- Adjust this value to simulate real-time behavior
    END LOOP;
    WAIT;
END PROCESS CLK_PROCESS;

STIMULUS_PROCESS : PROCESS
    VARIABLE seed1 : INTEGER := 666;
    VARIABLE seed2 : INTEGER := 69;

    IMPURE FUNCTION randomize_signal RETURN STD_LOGIC IS
        VARIABLE r : real;
        VARIABLE random_int : INTEGER;
        VARIABLE max : real := 1.0;
        VARIABLE min : real := 0.0;
    BEGIN
        uniform(seed1, seed2, r);
        random_int := INTEGER(r * (max - min)) + INTEGER(min);
        IF random_int = 0 THEN
            RETURN '0';
        ELSE
            RETURN '1';
        END IF;
    END FUNCTION;
```

Percobaan

Untuk menganalisis kinerja serta kesesuaian fungsi dari sistem Adaptive Traffic Light yang telah dibuat, maka praktikan melakukan ketepatan uji program dengan menggunakan komponen testbench. Dalam hal ini, testbench dirancang dengan menerapkan impure function pada VHDL untuk menghasilkan berbagai skenario sinyal sensor yang berbeda.

Secara garis besar, program testbench terdiri atas dua proses utama, yaitu generasi clock cycle serta value random untuk sinyal sensor. Clock cycle yang di generasi memiliki durasi cycle sepanjang satu detik, dimana low dan high state sama-sama bernilai 500ms. Dengan begitu, countdown timer dapat berjalan dengan durasi yang tepat dan akurat. Sementara itu, proses kedua bertujuan untuk menghasilkan 5 buah angka random untuk keempat sensor mobil dan satu tombol lansia. Adapun angka-angka random ini akan di generasi setiap 5 detik. Nilai 5 detik ini digunakan untuk merepresentasikan skenario update sinyal yang dapat saja terjadi di dunia nyata.

Hasil Percobaan



Analisis

Berdasarkan hasil simulasi yang dicantumkan pada Gambar 6, dapat dilihat bahwa ketika clock mengalami rising edge condition, maka sinyal road_vertical dan road_horizontal yang berperan dalam merepresentasikan warna lampu yang sedang menyala pada posisi terkait akan diinisialisasi dengan bit ‘000’, yang menandakan bahwa lampu belum aktif. Untuk kondisi rising edge yang kedua kalinya, maka program akan memasuki kondisi ‘INIT’ state, yang mana sinyal road_vertical akan diinisialisasikan dengan bit ‘100’ yang menandakan bahwa lampu merah akan aktif, sedangkan sinyal road_horizontal akan diinisialisaikan dengan bit ‘001’ yang menandakan bahwa lampu hijau akan aktif. Dalam hal ini, lamanya lampu merah dan lampu hijau akan aktif pada masing-masing sisi akan menyesuaikan waktu countdown yang direpresentasikan oleh signal display_vertical_one (untuk digit puluhan) dan signal display_vertical_two (untuk digit satuan). Adapun terms vertical dan horizontal pada signal display merepresentasikan posisi di mana countdown akan terjadi. Kondisi berikutnya ialah ketika terjadi rising edge pada saat kedua waktu countdown sudah mencapai nilai 0, maka signal road_vertical dan signal road_horizontal akan diinisialisasi dengan nilai bit ‘010’ yang menandakan bahwa lampu kuning akan aktif.

Kesimpulan

Pada dasarnya, implementasi kode VHDL ini menggambarkan sistem kontrol lampu lalu lintas adaptif yang responsif terhadap kondisi jalan yang berubah-ubah. Melalui entitas decoder, bilangan desimal diubah menjadi representasi Seven Segment yang diperlukan untuk menampilkan digit pada dua buah layar Seven Segment. Sedangkan entitas Adaptive Traffic Light menggunakan state machine dan logika kontrol yang rumit untuk mengatur perilaku lampu lalu lintas berdasarkan sejumlah variabel, seperti waktu, status tombol lansia, dan kondisi lalu lintas. Ini memungkinkan sistem untuk memperpanjang fase merah atau hijau berdasarkan situasi yang terdeteksi, seperti tombol lansia yang ditekan atau kepadatan lalu lintas yang tinggi di jalur tertentu.

Referensi

[1] “Data flow modeling :,” VHDL || Electronics Tutorial, https://www.electronics-tutorial.net/VHDL/Introduction/Data-Flow-Modeling/ (accessed Dec. 24, 2023).

[2] “VHDL behavioral modeling style,” Surf, https://surf-vhdl.com/vhdl-syntax-web-course-surf-vhdl/vhdl-behavioral-modeling-style/ (accessed Dec. 24, 2023).

[3] John, “How to write a basic testbench using VHDL,” FPGA Tutorial, https://fpgatutorial.com/how-to-write-a-basic-testbench-using-vhdl/ (accessed Dec. 24, 2023).

[4] “VHDL structural modeling style,” Surf, https://surf-vhdl.com/vhdl-syntax-web-course-surf-vhdl/vhdl-structural-modeling-style/ (accessed Dec. 24, 2023).

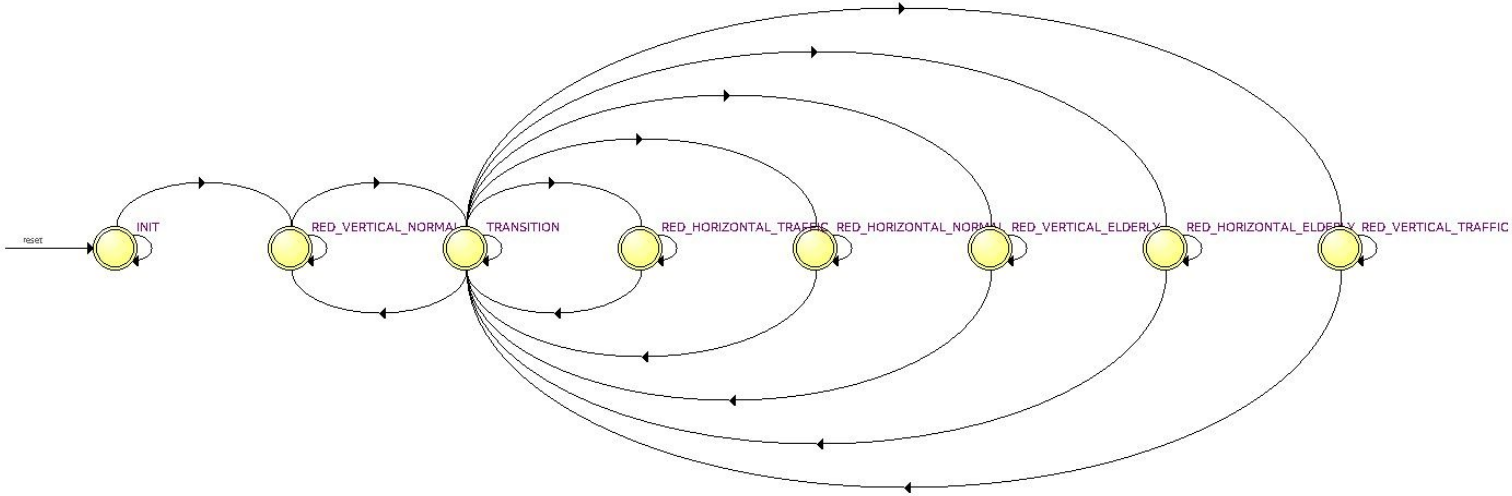
[5] Surf-VHDL, “VHDL for-loop statement - surf-VHDL,” Surf, https://surf-vhdl.com/vhdl-for-loop-statement/ (accessed Dec. 24, 2023).

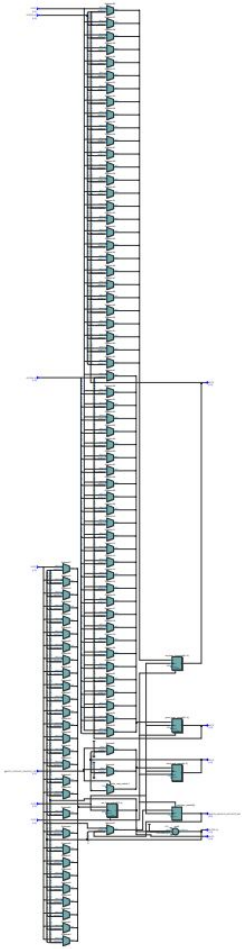
[6] J. J. Jensen, “How to use an impure function in VHDL,” VHDLwhiz, https://vhdlwhiz.com/impure-function/ (accessed Dec. 24, 2023).

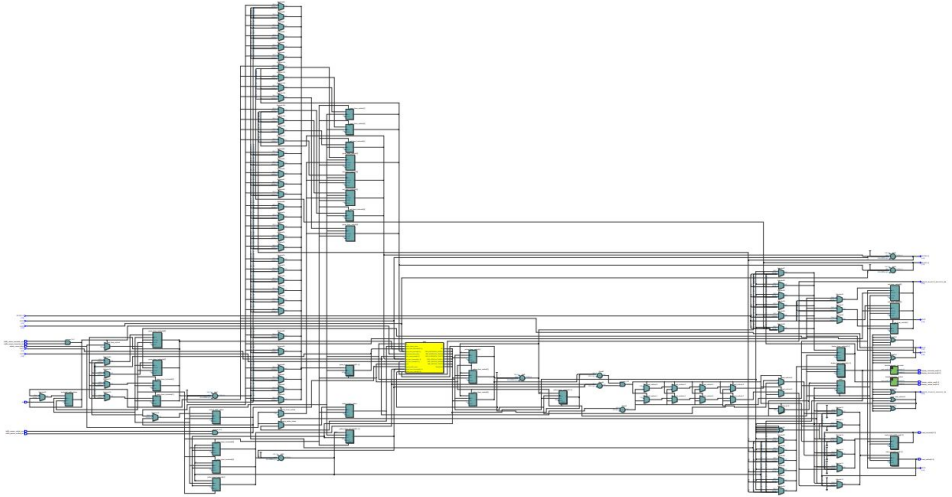
[7] J. J. Jensen, “How to create a finite-state machine in VHDL,” VHDLwhiz, https://vhdlwhiz.com/finite-state-machine/ (accessed Dec. 24, 2023).

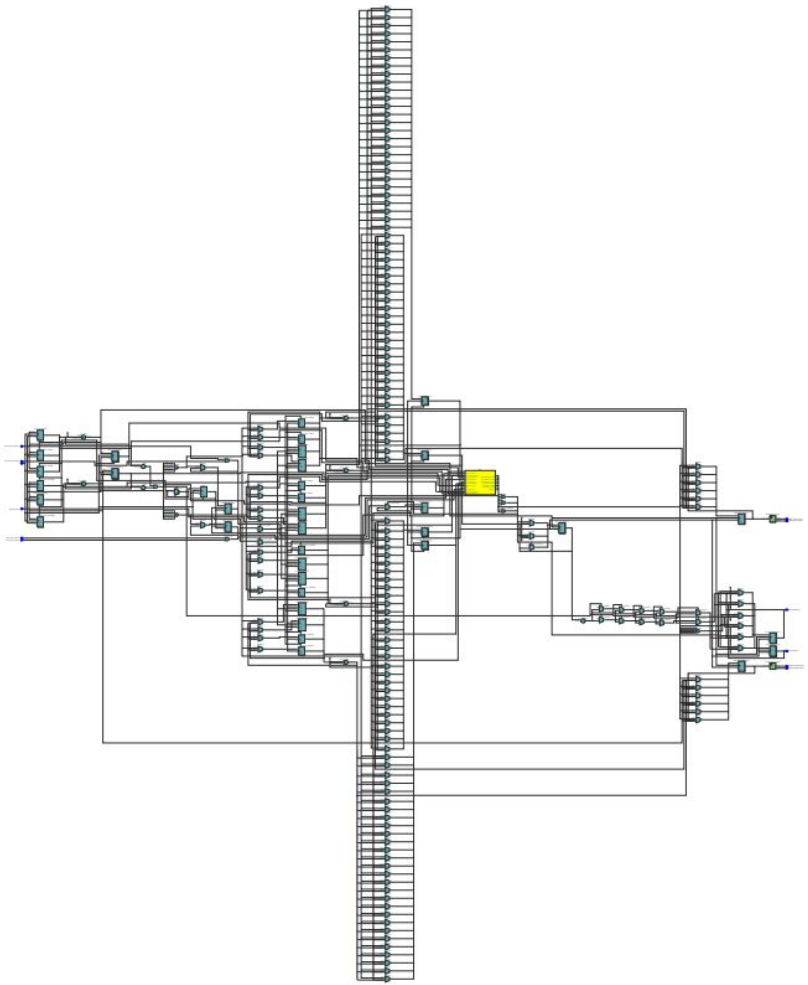
[8] Trace: Tennessee Research and Creative Exchange, https://trace.tennessee.edu/cgi/viewcontent.cgi?article=11034&context=utk_gradthes (accessed Dec. 24, 2023).

Appendix A: Project Schematic









Appendix B: Documentation