FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

FPGA-Based QR Code Encoder

GROUP PA03

Azra Nabila Azzahra	**2306161782**
Adhi Rajasa Rafif	**2306266943**
Adi Nugroho	**2306209095**

PREFACE

Pada Proyek Akhir PSD ini, kami merancang sebuah encoder QR Code berbasis VHDL (VHSIC Hardware Description Language) yang dapat menerima input 8 digit alphanumeric dan mengkonversikannya ke QR Code sesuai dengan standar encoding.

Dengan memanfaatkan konsep manipulasi data biner pada level bit, sistem ini dirancang untuk mengambil data masukan, memproses nya berdasarkan algoritma encoding QR Code, dan menghasilkan keluaran berupa pola QR Code yang dapat dibaca oleh perangkat pemindai. Proses encoding dimulai dengan mengonversi data masukan ke dalam format biner yang sesuai, di mana setiap bagian data diproses untuk memenuhi struktur dan aturan encoding QR Code. Pendekatan ini memungkinkan proses pengolahan data yang efisien, modular, dan akurat. Setelah proses konversi, sistem menghasilkan pola QR Code dalam bentuk matriks biner yang merepresentasikan data masukan dengan presisi tinggi.

Pendekatan berbasis VHDL memberikan fleksibilitas tinggi dalam mendesain sistem digital yang efisien dan dapat diimplementasikan pada perangkat keras seperti FPGA. Dengan metode ini, pengguna dapat memahami bagaimana data digital dapat direpresentasikan dalam bentuk visual yang sesuai dengan standar QR Code, memberikan wawasan tentang pengolahan data visual berbasis sistem digital.

Dalam laporan ini, kami akan menjelaskan secara rinci setiap tahap perancangan, mulai dari analisis kebutuhan, desain algoritma encoding, implementasi modul VHDL, hingga pengujian sistem.

Depok, December 06, 2024

Group PA03

TABLE OF CONTENTS

PREFACE.....2
CHAPTER 1: INTRODUCTION.....4
1.1 BACKGROUND.....4
1.2 PROJECT DESCRIPTION.....4
1.3 OBJECTIVES.....5
1.4 ROLES AND RESPONSIBILITIES.....5
CHAPTER 2: IMPLEMENTATION.....7
2.1 EQUIPMENT.....7
2.2 IMPLEMENTATION.....7
CHAPTER 3: TESTING AND ANALYSIS.....11
3.1 TESTING.....11
3.2 RESULT.....11
3.3 ANALYSIS.....12
CHAPTER 4: CONCLUSION.....13

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

QR Code merupakan teknologi modern yang digunakan untuk menyimpan dan mentransfer informasi secara cepat dan efisien. Teknologi ini telah diterapkan di berbagai bidang, seperti pembayaran digital, pemasaran, logistik, hingga autentikasi data. Kemampuan QR Code untuk mengemas informasi dalam format visual yang mudah dibaca menjadikannya salah satu solusi praktis dalam pengolahan data di era digital. Meski demikian, sistem pembuatan QR Code pada umumnya masih mengandalkan perangkat lunak berbasis prosesor tradisional, yang sering kali kurang optimal dari segi efisiensi dan kecepatan.

Dengan berkembangnya kebutuhan akan solusi yang lebih cepat dan fleksibel, FPGA atau Field-Programmable Gate Array muncul sebagai pilihan unggul. FPGA menawarkan keunggulan pemrosesan paralel dan fleksibilitas desain, memungkinkan pengembangan sistem digital yang efisien dan akurat. Dengan menggunakan VHDL atau VHSIC Hardware Description Language, pengembangan sistem berbasis FPGA dapat dilakukan secara modular dan presisi tinggi, menjadikannya alat yang ideal untuk merancang sistem seperti encoder QR Code yang handal dan sesuai standar.

1.2 PROJECT DESCRIPTION

Proyek ini bertujuan untuk merancang sistem encoder QR Code berbasis VHDL yang dapat menerima input alfanumerik hingga 8 karakter. Sistem akan mengonversi data input menjadi pola QR Code sesuai dengan standar encoding internasional, yang kemudian divisualisasikan dalam bentuk matriks biner. Proses encoding mencakup pengubahan data menjadi format biner, pengaturan elemen sesuai aturan QR Code, hingga pembangkitan pola yang dapat dibaca oleh perangkat pemindai.

Sistem ini memanfaatkan manipulasi data pada level bit untuk memastikan encoding yang efisien dan akurat. Setiap langkah dalam proses encoding dirancang modular, memanfaatkan keunggulan FPGA dalam pemrosesan paralel. Pendekatan ini memungkinkan pengolahan data real-time dengan hasil yang presisi, serta dapat diimplementasikan pada perangkat keras dengan performa tinggi.

Proyek ini juga mencakup pengujian melalui testbench untuk memastikan fungsi dan keakuratan sistem. Selain itu, penerapan konsep seperti FSM (Finite State Machine) digunakan untuk mengatur alur kerja sistem secara terstruktur, sehingga setiap tahap encoding berjalan sesuai dengan desain yang telah ditentukan. Proyek ini diharapkan memberikan pemahaman yang lebih mendalam tentang penerapan teknologi FPGA dalam pengolahan data visual berbasis QR Code.

1.3 OBJECTIVES

The objectives of this project are as follows:

1) Merancang encoder QR Code berbasis VHDL untuk input 8 karakter alfanumerik.
2) Menggunakan dataflow, behavioral, dan structural style programming dalam implementasi sistem.
3) Memanfaatkan looping construct serta procedure dan function dalam proses encoding.
4) Membuat FSM untuk mengatur tahapan encoding QR Code.
5) Mengembangkan testbench untuk memverifikasi sistem.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Pengembangan Kode	Pengolahan Input	Azra Nabila Azzahra
	Encoding	Azra Nabila Azzahra
	Pengolahan Error Generator	Azra Nabila Azzahra
	Perubahan txt ke bmp	Azra Nabila Azzahra
	Perubahan vhdl ke txt	Adhi Rajasa Rafif
Pembuatan Laporan	Pembuatan laporan pdf	Adhi Rajasa Rafif
	Pembuatan ReadMe	Azra Nabila Azzahra

	Pembuatan PPT	Adhi Rajasa Rafif

Table 1. Roles and Responsibilities

CHAPTER 2: IMPLEMENTATION

2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

- ModelSim
- VS Code
- QRCode Library
- Github

2.2 IMPLEMENTATION

Pada pengerjaan proyek ini, kami mengimplementasikan konsep-konsep dasar seperti Pemrograman VHDL, Concurrent Circuit Design, Sequential Circuit Design, Structural Circuit Design, Testbench, Looping Construct, dan Finite State Machine (FSM).

Data Encoder

Modul data_encoder menerima data dalam bentuk biner. Data teks berformat ASCII dengan panjang 8 karakter ini dikonversi menjadi alphanumeric menggunakan tabel LUT. Karakter akan dikelompokkan menjadi pasangan, dimana hasil akan dikonversi ke representasi biner 11 bit. Lantas, hasil encoding ini disusun dalam format mode indicator (4 bit), message length (8 bit), dan data (11 bit per pasangan).

LUTPackage

Merupakan tempat penyimpanan data statis, dalam hal ini tabel konversi karakter alfanumerik ke indeks. LUT menyimpan data dalam bentuk konstan atau fungsi akses. Dalam data_encoder, modul memanggil fungsi atau membaca konstan dari LUT untuk mendapatkan nilai yang sesuai.

Error Correction

Modul ini akan menerima data dari hasil Encoding (160 bit), dan memecahnya menjadi byte (8 bit). Dengan memanfaatkan algoritma Galois Field dan Generator Polynomial untuk menghitung nilai redudansi koreksi kesalahan (ECC), dihasilkanlah output berupa data ECC 2 byte.

Matrix Generator

Mengambil dua input, yakni data hasil encoding, dan data koreksi kesalahan. Lantas, ia akan membuat matriks kosong 21x21. Modul ini juga bertugas dalam pembuatan finder pattern dan timing pattern. Setelah itu, modul ini akan menyusun data dalam matriks, menghindari area yang sudah ditentukan untuk pola. Output adalah matriks QR yang direpresentasikan dalam bentuk vektor satu dimensi.

Masking

Akan mengambil output dari matrix generator dan melakukan proses mask pattern type 1 (i mod 2 == 0) untuk memvariasikan hasil.

Testbench

Digunakan untuk mengetes baik modul secara terpisah, ataupun saat menjadi satu.

Top-Level

Top-level bertindak sebagai penghubung semua modul dalam desain, dimana ia mengelola alur kerja antar modul menggunakan FSM.

IMPLEMENTASI PADA SETIAP MODUL

Dataflow Style Programming in VHDL

- Dataflow menggambarkan aliran data melalui elemen logika dalam desain. Dalam top_level, **dataflow** terlihat pada bagian output langsung dari sinyal tanpa banyak pemrosesan eksplisit, misalnya saat mentransmisikan sinyal seperti encoded_data atau ecc_data dari satu modul ke modul lain.

```
raw_matrix <= temp_matrix; (Output langsung tanpa manipulasi tambahan).
```

Behavioral Style Programming in VHDL

- Behavioral berfokus pada perilaku atau fungsi logika tanpa detail untuk implementasi fisiknya. Dalam program, ia umum digunakan. Contohnya pada data_encoder yang digunakan untuk menggambarkan bagaimana data diproses untuk menghasilkan output (teks diubah menjadi alphanumeric encoding).

Testbench

- Testbench dirancang untuk memverifikasi setiap modul dalam sistem. Input diberikan dalam bentuk string atau sinyal, dan output dibandingkan dengan hasil yang diharapkan menggunakan assert untuk memastikan akurasi implementasi. Contohnya adalah pengujian input seperti 'A' atau 'Z' untuk memeriksa apakah hasil encoding sesuai standar QR Code.

Structural Style Programming in VHDL

- Digunakan untuk menyusun desain menggunakan komponen yang saling dihubungkan. Pada Top-Level Module, yang mengintegrasikan beberapa komponen seperti data_encoder, error_correction, dan modul lainnya. Pendekatan ini memastikan desain bersifat modular, sehingga memudahkan debugging dan pengembangan lebih lanjut.

Looping Construct

- Merupakan struktur kontrol yang memungkinkan pengulangan operasi. Diterapkan pada banyak modul, terutama dalam penyusunan matriks. Dalam data_encoder, looping digunakan untuk memproses pasangan karakter, di matrix_generator, digunakan untuk menempatkan path finder dan timing.

Procedure, Function, and Impure Function

- Merupakan blok logika terpisah untuk mempermudah pengulangan operasi tertentu. penerapannya dapat dilihat pada to_binary_vector yang digunakan dalam data_encoder untuk mengkonversi alphanumeric ke biner. Terdapat pula fungsi gf_mult dalam error_correction untuk multiplikasi Galois Field. Fungsi ini menyediakan konversi yang efisien dan reusable dalam berbagai modul.

Finite State Machine (FSM)

- FSM merupakan model kontrol yang mengatur transisi antar kondisi untuk mengelola operasi logika kompleks. Dalam penerapannya, dapat dilihat pada top_level yang digunakan untuk **mengontrol alur data antar modul** dan **menangani tahapan penyusunan QR Code**. Dimana ia akan mengambil input (dalam idle state), melakukan encoding, ecc, matrix, dan terakhir menyelesaikan proses.

CHAPTER 3: TESTING AND ANALYSIS

3.1 TESTING

Pengujian sistem encoder QR Code dilakukan untuk memastikan seluruh proses berjalan sesuai dengan spesifikasi, mulai dari penerimaan input string hingga menghasilkan file gambar QR Code yang dapat dibaca. Pengujian ini bertujuan untuk memverifikasi keakuratan proses encoding, integrasi antar modul, serta validasi hasil output akhir.

Pada pengujian, input berupa string "HALLO123" diberikan melalui simulasi di ModelSim. Sistem diuji menggunakan file testbench yang telah dirancang untuk memverifikasi setiap langkah, termasuk konversi string menjadi integer, representasi biner, dan penggabungan data. Hasil representasi biner ditulis ke file teks (.txt) untuk diuji lebih lanjut. File teks ini kemudian diproses menggunakan perangkat lunak tambahan untuk menghasilkan file gambar QR Code dalam format .bmp. File gambar yang dihasilkan diuji menggunakan pemindai QR Code untuk memastikan data yang terkandung sesuai dengan input asli. Pengujian juga mencakup skenario error untuk memastikan sistem dapat menangani input tidak valid dengan baik.

3.2 RESULT

Hasil pengujian menunjukkan bahwa sistem encoder QR Code berhasil mengonversi input string "HALLO123" menjadi representasi alphanumeric yang sesuai. Representasi ini diekspor ke file teks (.txt), yang kemudian diproses secara manual menggunakan perangkat lunak tambahan untuk menghasilkan file gambar QR Code dalam format .bmp. File gambar yang dihasilkan diuji menggunakan pemindai QR Code, yang menunjukkan bahwa data asli, yaitu "HALLO123".

Meskipun proses encoding QR Code hingga representasi biner bekerja sesuai harapan, konversi dari file teks ke gambar QR Code belum jalan di dalam kode. Oleh karena itu, tahap akhir pembuatan file gambar QR Code dilakukan secara manual. Hasil akhir berupa gambar QR Code yang berhasil dibaca oleh pemindai hp menunjukkan bahwa sistem telah bekerja seperti dengan tujuan proyek, meskipun pada kodenya masih belum sempurna dan masih terdapat kesalahan.

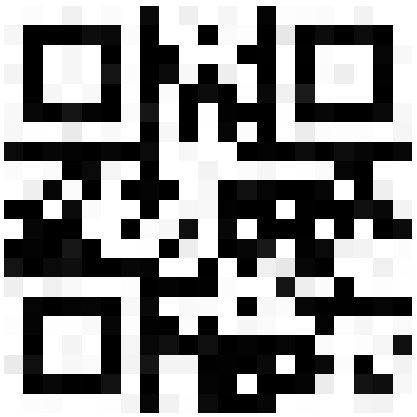


Fig 1. QR Code

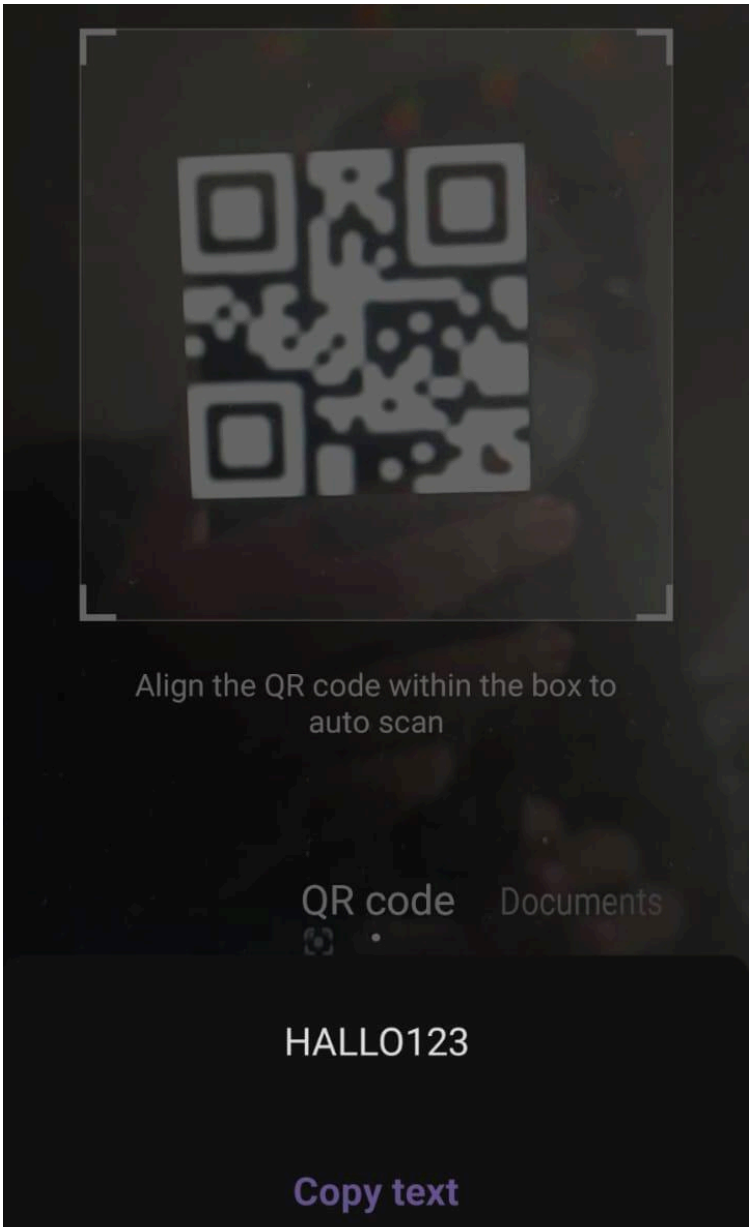


Fig 2. Testing Result

3.3 ANALYSIS

Sistem encoder QR Code berhasil mengonversi input string menjadi representasi biner yang sesuai dan menuliskannya ke file teks. Namun, implementasi sistem belum sempurna, karena tahap konversi dari file teks ke gambar .bmp belum berjalan otomatis. Hal ini menyebabkan proses akhir harus dilakukan secara manual menggunakan perangkat lunak tambahan. Meskipun ada keterbatasan tersebut, hasil akhir berupa QR Code yang dapat dibaca menunjukkan bahwa algoritma encoding telah berjalan dengan benar, namun perlu perbaikan pada bagian integrasi otomatisasi.

CHAPTER 4: CONCLUSION

Sistem encoder QR Code yang dirancang pada proyek ini telah mampu mengonversi input string menjadi representasi biner sesuai standar encoding QR Code. Proses encoding hingga tahap penulisan ke file teks berjalan dengan baik, dan hasilnya dapat digunakan untuk menghasilkan QR Code yang dapat dibaca menggunakan perangkat pemindai. Meskipun sistem belum mampu menghasilkan file gambar .bmp secara otomatis, konversi manual menunjukkan bahwa data input dapat direpresentasikan dengan benar dalam bentuk QR Code.

Kendala pada tahap akhir konversi menunjukkan perlunya perbaikan pada integrasi sistem untuk mendukung otomatisasi penuh. Secara keseluruhan, proyek ini telah berhasil menunjukkan implementasi algoritma encoding QR Code menggunakan VHDL, meskipun masih ada ruang untuk pengembangan lebih lanjut, terutama pada aspek otomatisasi dan efisiensi proses. Hasil proyek ini diharapkan dapat menjadi dasar bagi pengembangan sistem encoding QR Code yang lebih lengkap dan terintegrasi di masa depan.

REFERENCES

[1] "*QR-Code Generator Based on FPGA (pdf) | Paperity*," *Paperity.org*, 2020. https://paperity.org/p/292788494/qr-code-generator-based-on-fpga (accessed Nov. 20, 2024).

[2] "*QR Code Tutorial - Thonky.com*," *www.thonky.com*. https://www.thonky.com/qr-code-tutorial/ (accessed Nov. 20, 2024).

[3] TienDungHUST, "*GitHub - TienDungHUST/Encode-QR-code-using-VHDL: Use VHDL to generate QR which contain 6 digits*," *GitHub*, 2021. https://github.com/TienDungHUST/Encode-QR-code-using-VHDL (accessed Nov. 28, 2024).

[4] "*QRazyBox - Help Page,*" *Merri.cx*, 2024. https://merri.cx/qrazybox/help/getting-started/about-qr-code.html (accessed Nov. 19, 2024).

[5] I. A. http ://www.injosoft.se, “ASCII Code - The extended ASCII table,” *www.ascii-code.com*, 2005. https://www.ascii-code.com/ (accessed Dec. 2, 2024).

APPENDICES

Appendix A: Project Schematic

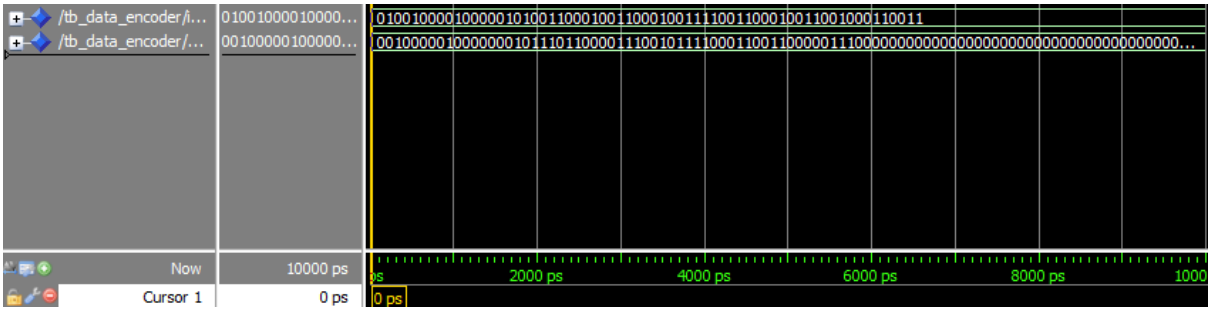


Fig. Waveform data_encoder

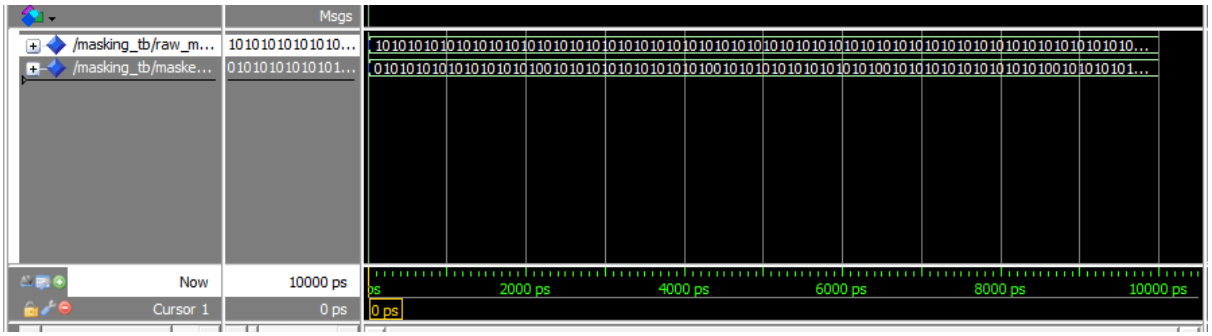


Fig. Waveform masking

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

use work.LUTPackage.ALL;


entity Top_Level is
```

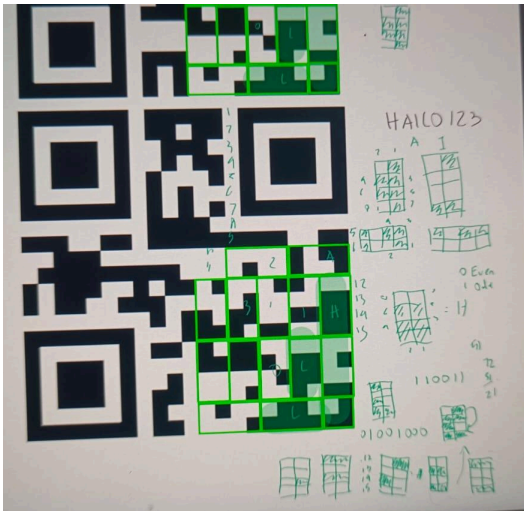
```
    Port (
        clk         : in std_logic;
        reset       : in std_logic;
        start       : in std_logic;
        input_text  : in std_logic_vector(63 downto 0);
        done        : out std_logic;
        qr_matrix   : out std_logic_vector(21*21-1 downto 0)
    );
end Top_Level;

architecture Behavioral of Top_Level is
    -- FSM States
    type state_type is (IDLE, ENCODING, ECC_GENERATION,
MATRIX_GENERATION, MASKING, OUTPUT);
    signal state, next_state : state_type;

    -- Internal signals
    signal encoded_data  : std_logic_vector(159 downto 0);
    signal ecc_data      : std_logic_vector(39 downto 0);
    signal raw_matrix    : std_logic_vector(21*21-1 downto 0);
    signal masked_matrix : std_logic_vector(21*21-1 downto 0);
    signal write_ready   : std_logic;

begin
    -- State transition logic
    process(clk, reset)
    begin
        if reset = '1' then
            state <= IDLE;
```

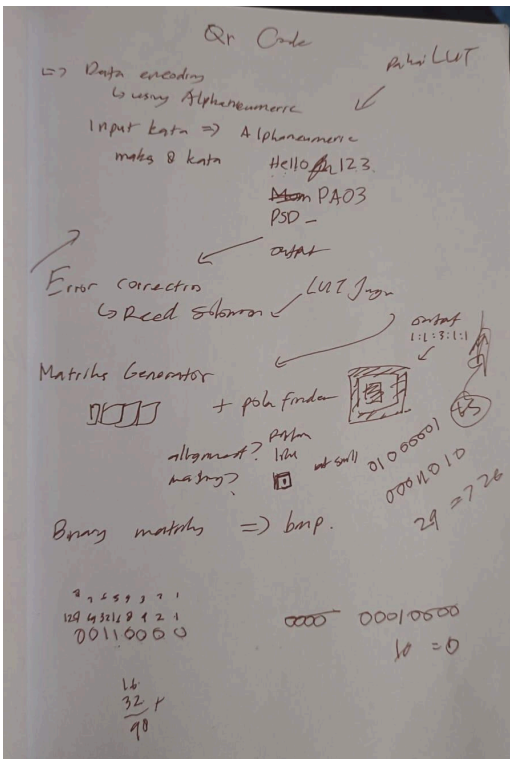
```
        elsif rising_edge(clk) then
            state <= next_state;
        end if;
    end process;

    process(state, start, write_ready)
    begin
        case state is
            when IDLE =>
                if start = '1' then
                    next_state <= ENCODING;
                else
                    next_state <= IDLE;
                end if;

            when ENCODING =>
                encoder_inst : data_encoder port map(
                    input_text => input_text,
                    encoded_data => encoded_data
                );
                next_state <= ECC_GENERATION;

            when ECC_GENERATION =>
                ecc_inst : error_correction port map(
                    encoded_data => encoded_data,
                    ecc_data => ecc_data
                );
                next_state <= MATRIX_GENERATION;
```

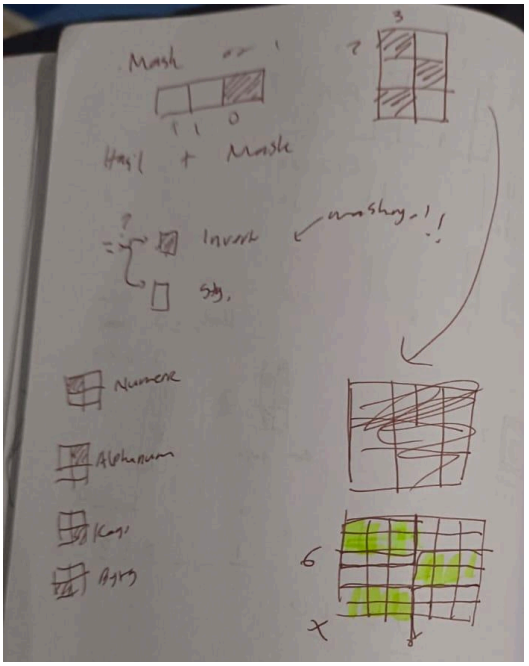
```
            when MATRIX_GENERATION =>
                -matrix_inst : matrix_generator port map(
                    encoded_data => encoded_data,
                    ecc_data => ecc_data,
                    raw_matrix => raw_matrix
                );
                next_state <= MASKING;

            when MASKING =>
                masking_inst : masking port map(
                    raw_matrix => raw_matrix,
                    masked_matrix => qr_matrix
                );
                next_state <= OUTPUT;

            when OUTPUT =>
                if write_ready = '1' then
                    next_state <= IDLE;
                else
                    qr_matrix <= qr_matrix;
                    next_state <= OUTPUT;
                end if;

            when others =>
                next_state <= IDLE;
        end case;
    end process;

    -- Output logic
```

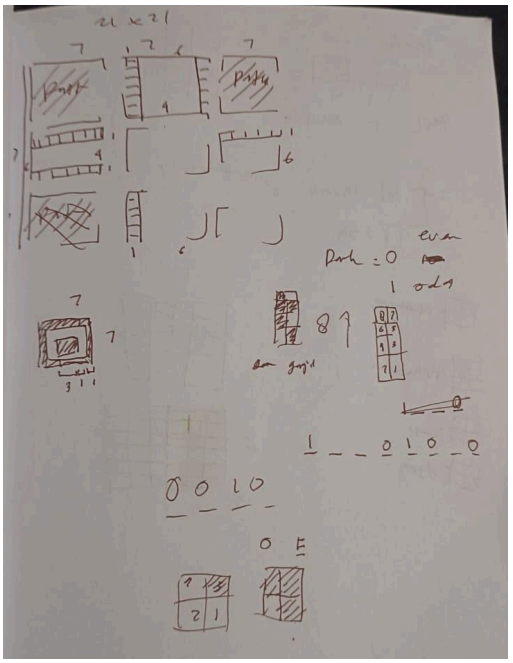
```
    done <= '1' when state = OUTPUT else '0';

    qr_matrix <= masked_matrix;


end Behavioral;
```

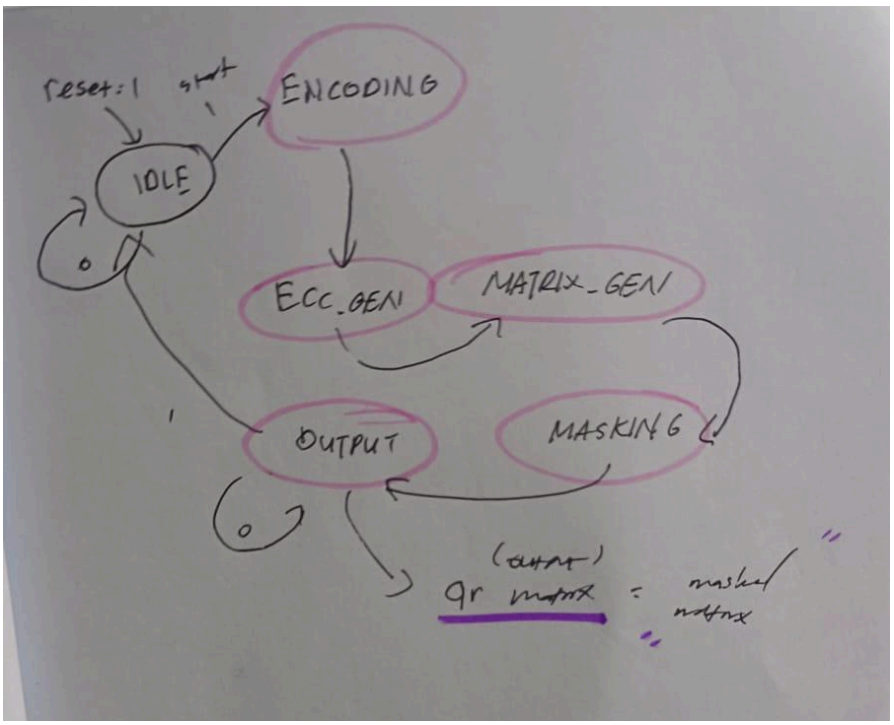
Appendix B: Documentation




Qr Code
Data encoding
using Alphanumeric
Input kata => Alphanumeric
maks 8 kata
Hello123
PA03
PSD
output
Error correction
Reed Solomon
LUT Jogja
Matriks Generator
+ pola finder
alignment?
masking?
Bentuk matriks => bmp.
29 = 726


Mask
Hasil + Mask
invert
masking!!
Numeric
Alphanum
Kanji
Byte


Parity
Parity
Parity = 0
even
1 odd


reset=1
start
1
ENCODING
IDLE
0
ECC_GEN
MATRIX_GEN
1
OUTPUT
MASKING
0
(output)
qr matrix
=
masked matrix