LAPORAN PROYEK AKHIR PRAKTIKUM PERANCANGAN SISTEM DIGITAL
(RSA ENCRYPTOR)

Group BP10:

Muhammad Fahish Haritsah Bimo	2206059616
Raja Yonandro Ruslito	2206059553
Yoel Dwi Miryano	2206059534
Zikri Zulfa Azhim	2206028390

Asisten Laboratiorium:

Eldisja Hadasa

Perancangan Sistem Digital

Program Studi Teknik Komputer

Departemen Teknik Elektro

Fakultas Teknik

Universitas Indonesia

KATA PENGANTAR

Puji dan syukur kita panjatkan kehadirat Tuhan YME yang telah memberikan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan proyek akhir Praktikum Perancangan Sistem Digital yang berjudul “RSA Encryptor”.

Perkembangan teknologi yang berkembang selama ini membuat teknologi menjadi suatu hal yang tidak bisa dilepaskan dalam kehidupan sehari-hari karena teknologi sudah masuk dalam kehidupan kita. Proyek akhir ini diharapkan bisa menjadi salah satu terapan dalam perkembangan teknologi saat ini dan bisa dikembangkan lebih lanjut sehingga bisa bekerja lebih baik kelak.

Kami mengucapkan terima kasih kepada anggota kelompok BP10 yang membantu dalam penyusunan laporan ini sehingga laporan ini dapat diselesaikan. Ucapan terima kasih juga disampaikan kepada Kak Eldisja Hadasa selaku pembimbing kelompok BP10 yang telah mendukung dan membantu dalam pengerjaan sehingga kami bisa menyelesaikan proyek ini dengan baik.

Kami menyadari bahwa proyek ini masih memiliki kekurangan. Oleh karena itu, saran dan kritik sangat diharapkan untuk membuat kami lebih baik lagi kedepannya. Semoga laporan ini bisa bermanfaat dalam dunia pendidikan, khususnya di bidang teknologi.

DAFTAR ISI

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Dalam era digital saat ini, keamanan informasi menjadi hal yang sangat penting. Banyak data sensitif dan pribadi yang ditransmisikan melalui jaringan komunikasi, seperti pesan teks, email, dan berbagai jenis file. Oleh karena itu, perlindungan terhadap data tersebut menjadi suatu kebutuhan yang mendesak.

Namun, masalah utama yang dihadapi adalah potensi serangan oleh pihak-pihak yang tidak berwenang yang berusaha mencuri atau merusak data. Teknik paling umum yang digunakan oleh para peretas adalah mencoba mendekripsi informasi yang dikirimkan atau disimpan. Untuk mengatasi masalah ini, diperlukan suatu sistem enkripsi yang kuat dan efisien.

1.2. Solusi

Program enkripsi merupakan suatu solusi yang efektif untuk melindungi kerahasiaan dan integritas data. Enkripsi melibatkan proses mengubah informasi asli menjadi bentuk yang tidak dapat dimengerti tanpa memiliki kunci dekripsi yang sesuai. Berikut beberapa langkah solusi yang dapat diambil:

a. Penggunaan Algoritma Enkripsi Kuat:
   Memilih algoritma enkripsi yang handal dan aman sangat penting. Algoritma seperti AES (Advanced Encryption Standard) telah terbukti kuat dan menjadi standar industri untuk melindungi data.
b. Manajemen Kunci yang Efisien:
   Kunci enkripsi harus dikelola dengan cermat. Penggunaan kunci yang kuat dan siklus pemberian kunci yang teratur dapat meningkatkan keamanan sistem.
c. Implementasi Protokol Keamanan Komunikasi:

Dalam komunikasi jaringan, protokol keamanan seperti HTTPS (SSL/TLS) dapat digunakan untuk melindungi data saat berpindah melalui jaringan.

d. Pembaruan dan Pemeliharaan Teratur:
   Sistem enkripsi harus diperbarui secara berkala untuk mengatasi kerentanan baru yang mungkin muncul. Pemeliharaan rutin termasuk pembaruan algoritma enkripsi dan perangkat lunak terkait.
e. Pelatihan Pengguna:
   Pemahaman yang baik tentang pentingnya enkripsi dan keamanan data perlu ditanamkan pada pengguna. Pelatihan dapat membantu mereka memahami cara menggunakan dan melibatkan sistem enkripsi dengan benar.
f. Audit Keamanan secara Berkala:
   Melakukan audit keamanan secara teratur dapat membantu mengidentifikasi potensi kerentanan dan memastikan kepatuhan terhadap standar keamanan.

Dengan mengimplementasikan program enkripsi yang kokoh, dapat diharapkan bahwa kerahasiaan dan integritas data dapat terjaga dengan baik, sehingga risiko ancaman keamanan dapat diminimalkan.

1.3. Pembagian Tugas

Untuk pengerjaan proyek ini, kelompok kami melakukan pembagian tugas menjadi sebagai berikut:

Muhammad Fahish Haritsah Bimo	: Readme.md
Raja Yonandro Ruslito	: Laporan
Yoel Dwi Miryano	: Design Hardware
Zikri Zulfa Azhim	: PPT

BAB II

PEMBAHASAN

2.1. Peralatan

Peralatan yang digunakan untuk pengerjaan proyek ini, antara lain:

a. Virtual Studio Code sebagai IDE untuk coding.
b. Github sebagai tempat untuk sharing code yang sudah dibuat. (https://github.com/)
c. ModelSim sebagai simulator
d. Draw.io sebagai ilustrasi diagram. (https://app.diagrams.net/)
e. Quartus Prime

2.2. Implementasi

Program ini dapat membuat random number menggunakan impure function, yang terdapat di RandomPrimeGenerator.vhd. Pada program tersebut, akan ter-generate dua random number 12 bit (4096 dalam desimal) dan akan dilakukan pengecekan apakah bilangan tersebut prima atau tidak, kalua didapatkan kedua bilangan tersebut random, maka kedua bilangan tersebut akan dimasukkan ke dalam **p** dan **q**.

Kemudian cara kerja dari program ini sendiri adalah sebagai berikut:

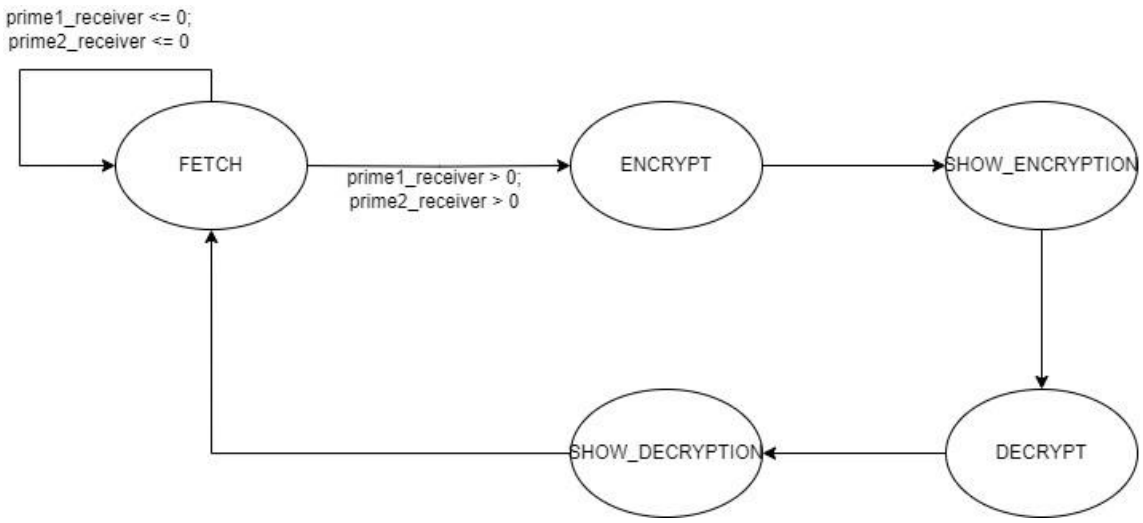
1) Pembuatan Kunci:
   - Langkah 1: Alice (sebagai Sender) ingin berkomunikasi secara aman dengan Bob (Ssebagai Receiver) melalui internet. Perangkat Alice pertama-tama membuat sepasang kunci, kunci publik dan kunci privat.
   - Langkah 2: Perangkat Alice memilih dua bilangan prima yang besar secara acak, misalnya p dan q. Kemudian, akan menghitung hasil perkalian kedua bilangan tersebut, yaitu n = p * q. Ini akan menjadi bagian dari kunci publik dan juga kunci privat.

- Langkah 3: Selanjutnya, perangkat Alice memilih bilangan lain, misalnya e, yang merupakan bilangan bulat positif yang relatif prima terhadap (p-1) * (q-1). Pasangan (n, e) menjadi kunci publik dari perangkat Alice.

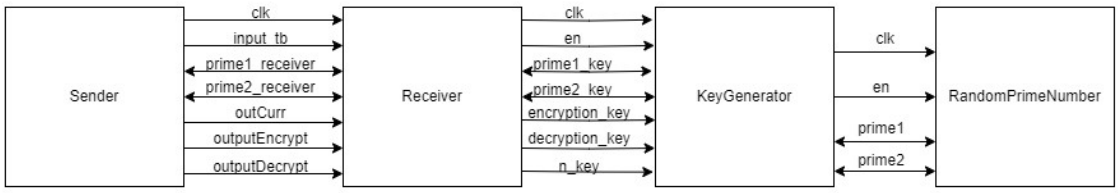
2) Enkripsi Pesan:
   - Langkah 1: Bob ingin mengirim pesan rahasia kepada Alice. Pesan tersebut diubah menjadi bilangan bulat di dalam perangkat Bob, misalnya m.
   - Langkah 2: Perangkat Bob menggunakan kunci publik dari perangkat Alice (n, e) untuk mengenkripsi pesannya. Operasi enkripsi dilakukan dengan rumus matematis yang mengubah pesan m menjadi teks terenkripsi, misalnya c.
   - Langkah 3: Perangkat Bob mengirim teks terenkripsi c kepada perangkat Alice melalui saluran komunikasi yang tidak aman.

3) Dekripsi Pesan:
   - Langkah 1: Perangkat Alice menerima teks terenkripsi c dari perangkat Bob.
   - Langkah 2: Perangkat Alice menggunakan kunci privat yang hanya dimilikinya, yaitu bilangan bulat d, untuk mendekripsi pesan. Bilangan d dipilih sedemikian rupa sehingga d * e (dalam modulo (p-1) * (q-1)) sama dengan 1.
   - Langkah 3: Dengan menggunakan rumus matematis, perangkat Alice melakukan operasi dekripsi pada teks terenkripsi c dengan menggunakan kunci privatnya, dan mendapatkan kembali pesan asli m.

Proses ini memastikan bahwa meskipun teks terenkripsi c dapat ditransmisikan melalui saluran komunikasi yang tidak aman, hanya penerima yang memiliki kunci privat yang cocok yang dapat mendekripsi pesan tersebut kembali ke dalam bentuk aslinya. Dalam hal ini, perangkat Alice adalah satu-satunya yang memiliki kunci privat untuk mendekripsi pesan yang dikirim oleh perangkat Bob. Ini memastikan kerahasiaan pesan antara Alice dan Bob dalam komunikasi mereka. Berikut adalah state diagram dan block diagram dari program kami:



Gambar 1. State Diagram Program



Gambar 2. Block Diagram Program

2.2.1. Sender

Kode ini adalah sebuah program testbench untuk program utama kami, yaitu Reciever.vhd. Kode VHDL tersebut menggambarkan entitas ("entity") yang disebut "Sender" dan sebuah arsitektur ("architecture") yang terkait dengannya. Entitas Sender memiliki input berupa sinyal clock (**clk**), dua input bertipe bilangan bulat (**prime1_receiver** dan **prime2_receiver**), serta tiga output bertipe karakter (**outCurr**, **outputEncrypt**, dan **outputDecrypt**). Selain itu, entitas ini menginstansiasi sebuah komponen ("component") yang disebut "Receiver".

Arsitektur dari entitas Sender mencakup deklarasi komponen Receiver, serta sebuah sinyal bernama **input_tb** yang diinisialisasi dengan nilai string. Terakhir, terdapat sebuah instansiasi komponen Receiver yang terhubung dengan input dan output yang sesuai.

2.2.2. Receiver

Kode VHDL ini mendefinisikan sebuah entitas ("entity") yang disebut "Receiver" bersama dengan arsitekturnya. Entitas ini memiliki beberapa port input dan output, termasuk sinyal clock (**clk**), string input (**input**), dua buffer integer (**prime1_receiver** dan **prime2_receiver**), serta tiga karakter output (**outCurr**, **outputEncrypt**, dan **outputDecrypt**).

Arsitektur "rtl" dari entitas Receiver ini menggunakan sebuah komponen ("component") yang disebut "KeyGenerator". Komponen ini memiliki beberapa port input dan output, termasuk clock (**clk**), enable (**en**), dua buffer integer (**prime1_key** dan **prime2_key**), serta tiga integer output (**encryption_key**, **decryption_key**, dan **n_key**).

Selain itu, terdapat deklarasi beberapa sinyal yang digunakan untuk mengimplementasikan sebuah state machine. Sinyal **currstate** dan **nextstate** merupakan jenis enumerasi (**state**) yang menyimpan state saat ini dan state selanjutnya. Ada juga sinyal **en_receiver** yang digunakan untuk mengaktifkan atau menonaktifkan Receiver.

Proses pertama (**process(clk)**) mengimplementasikan state machine untuk melakukan enkripsi dan dekripsi. Proses ini diaktifkan pada setiap naiknya sinyal clock (**rising_edge(clk)**). Proses ini juga mencakup beberapa variable dan case statement untuk mengatur transisi antar state. Setiap state melibatkan operasi matematika terkait enkripsi dan dekripsi RSA.

Proses kedua (**process(clk)**) mengupdate state saat ini (**currstate**) dengan state selanjutnya (**nextstate**) pada setiap naiknya sinyal clock. Ini digunakan untuk menggerakkan state machine.

Secara keseluruhan, entitas Receiver ini terlihat sebagai implementasi state machine untuk enkripsi dan dekripsi RSA menggunakan komponen KeyGenerator.

State machine ini mengatur alur data dan kontrol operasi enkripsi dan dekripsi untuk setiap karakter input pada string **input**.

2.2.3. Key Generator

Program VHDL ini mengimplementasikan entitas "Key Generator" yang bertanggung jawab menghasilkan kunci publik dan kunci privat berdasarkan bilangan prima **p_key** dan **q_key**. Komponen utama dalam program ini adalah proses yang teraktivasi pada setiap edge naik dari sinyal clock (**clk**). Dalam proses tersebut, terdapat komponen lain yang disebut **RandomPrimeGenerator**. Komponen ini bertanggung jawab untuk menghasilkan dua bilangan prima yang akan menjadi **p_key** dan **q_key**.

Selanjutnya, di dalam proses utama, dilakukan perhitungan untuk menentukan kunci enkripsi (**e**) dan kunci dekripsi (**d**) berdasarkan algoritma yang tertera. Hasil perhitungan ini kemudian disimpan dalam sinyal **public_key** dan **private_key**. Program ini juga memberikan komentar yang menunjukkan tempat untuk menambahkan logika tambahan, seperti perhitungan kunci privat (**d**), yang mungkin diperlukan dalam implementasi kriptografi yang lebih kompleks. Singkatnya, tujuan program ini adalah menghasilkan kunci publik dan privat yang digunakan dalam algoritma kriptografi, di mana kunci publik digunakan untuk enkripsi dan kunci privat untuk dekripsi.

2.2.4. Random Prime Generator

Pada Program "RandomPrimeGenerator", dirancang untuk menghasilkan dua bilangan prima (**p** dan **q**) berdasarkan sinyal clock (**clk**). Program ini menggunakan metode uji coba (trial division) untuk memastikan bahwa bilangan yang dihasilkan adalah prima. Dalam implementasinya, program menggunakan proses yang diaktifkan pada setiap edge naik dari sinyal clock. Di dalam proses tersebut, dilakukan inisialisasi variabel-variabel, seperti **seed1**, **seed2**, **rand_int1**, dan **rand_int2**, yang diperlukan untuk menghasilkan bilangan acak.

Program ini memiliki fungsi **rand_int_gen** yang menghasilkan bilangan bulat acak dalam rentang tertentu. Fungsi ini menggunakan nilai-nilai seed dan fungsi uniform dari paket **IEEE.math_real** untuk menghasilkan nilai acak antara 0 dan 1, kemudian diubah menjadi bilangan bulat dalam rentang yang ditentukan.

Dalam setiap siklus proses, program menghasilkan dua bilangan acak (**rand_int1** dan **rand_int2**). Selanjutnya, program menggunakan metode uji coba (trial division) untuk memeriksa apakah keduanya adalah bilangan prima. Proses ini dilakukan dengan membagi masing-masing bilangan dengan angka-angka dari 2 hingga akar kuadrat dari bilangan tersebut. Jika ditemukan pembagi yang membagi habis, maka bilangan tersebut bukan prima.

Proses ini diulang hingga kedua bilangan yang dihasilkan benar-benar adalah bilangan prima. Setelah itu, bilangan prima tersebut disalin ke dalam output **p** dan **q**. Ringkasnya, tujuan dari program ini adalah memberikan dua bilangan prima (**p** dan **q**) sebagai output berdasarkan sinyal clock, dengan memastikan bahwa bilangan-bilangan tersebut benar-benar adalah bilangan prima melalui metode uji coba.

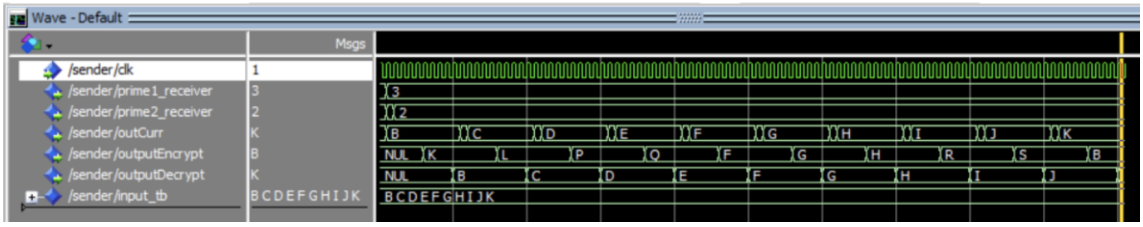
BAB III

TEST DAN ANALISA PROGRAM

3.1. Hasil Pengujian

TESTBENCH

- Sender.vhd



Gambar 3. Ouput Program Testbench Sender.vhd

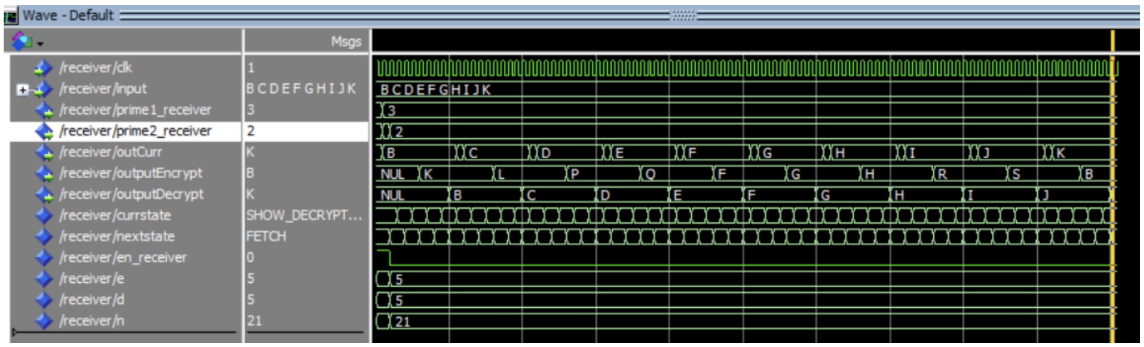
Source Code:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity Sender is
    port (
        clk: in std_logic;
        prime1_receiver, prime2_receiver: buffer integer;
        outCurr: out character;
        outputEncrypt: out character;
        outputDecrypt: out character
    );
end entity Sender;

architecture rtl of Sender is
    component Receiver is
        port (
            clk: in std_logic;
            input: in string (1 to 19);
            prime1_receiver, prime2_receiver: buffer integer;
            outCurr: out character;
            outputEncrypt: out character;
            outputDecrypt: out character
        );
```

```
    end component Receiver;

    signal input_tb: string(1 to 19) := “B C D E F G H I J K”;
begin
    Receiver1: Receiver port map (
        clk => clk,
        input => input_tb,
        prime1_receiver => prime1_receiver,
        prime2_receiver => prime2_receiver,
        outCurr => outCurr,
        outputEncrypt => outputEncrypt,
        outputDecrypt => outputDecrypt
    );
end architecture rtl;
```

MAIN PROGRAM

- Receiver.vhd



Gambar 4. Output Program Receiver.vhd

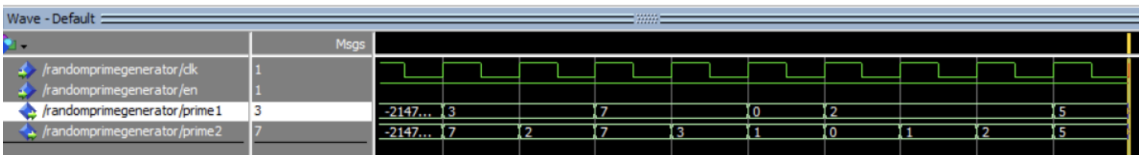
Source Code:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity Receiver is
    port (
        clk: in std_logic;
        input: in string (1 to 19) := "B C D E F G H I J K";
        prime1_receiver, prime2_receiver: buffer integer;
        outCurr: out character;
        outputEncrypt: out character;
        outputDecrypt: out character
```

```
    );
end entity Receiver;

architecture rtl of Receiver is

    component KeyGenerator is
        port (
            clk: in std_logic;
            en: in std_logic;
            prime1_key, prime2_key: buffer integer;
            encryption_key, decryption_key, n_key: out integer
        );
    end component KeyGenerator;

    --STATES
    type state is (FETCH, ENCRYPT, SHOW_ENCRYPTION, DECRYPT,
SHOW_DECRYPTION);

    --SIGNAL FOR STATES
    signal currstate, nextstate: state;

    --ENABLE SIGNAL
    signal en_receiver: std_logic := '1';
    signal e, d, n: integer;

begin
    KeyGen: KeyGenerator port map (
        clk => clk,
        en => en_receiver,
        prime1_key => prime1_receiver,
        prime2_key => prime2_receiver,
        encryption_key => e,
        decryption_key => d,
        n_key => n
    );

    process(clk)
        --TEMPORARY VARIABLES TO STORE THE INTEGER OF A CHAR
        variable i_int, e_int, d_int: integer := 0;
        variable pc: integer := 1;
    begin

        if rising_edge(clk) then
            outCurr <= input(pc);
            case currstate is
```

```
                when FETCH =>
                    i_int := character'pos(input(pc)) - 64;

                    if prime1_receiver > 0 AND prime2_receiver > 0 then
                        en_receiver <= '0';
                        nextstate <= ENCRYPT;
                    end if;
                when ENCRYPT =>
                    e_int := (i_int ** e) mod n;
                    nextstate <= SHOW_ENCRYPTION;
                when SHOW_ENCRYPTION =>
                    outputEncrypt <= character'val(e_int + 64);
                    nextstate <= DECRYPT;
                when DECRYPT =>
                    d_int := (e_int ** d) mod n;
                    nextstate <= SHOW_DECRYPTION;
                when SHOW_DECRYPTION =>
                    outputDecrypt <= character'val(d_int + 64);
                    pc := pc + 1;
                    nextstate <= FETCH;
            end case;
        end if;
    end process;

    process(clk)
    begin
        if rising_edge(clk) then
            currstate <= nextstate;
        end if;
    end process;


end architecture rtl;
```

- RandomPrimeGenerator.vhd



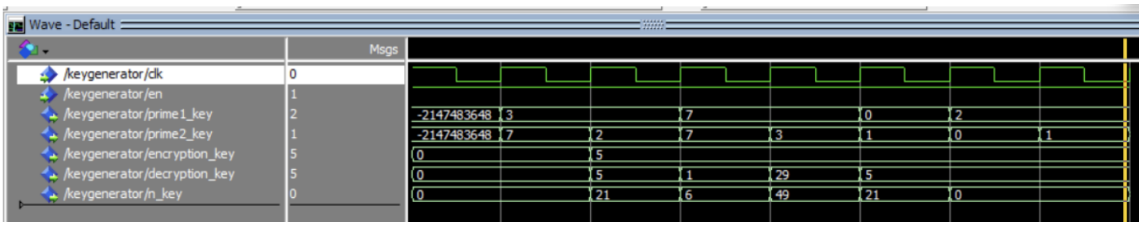
Gambar 5. Output Program RandomPrimeGenerator.vhd

Source Code:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.math_real.all;

entity RandomPrimeGenerator is
    port (
        clk: in std_logic;
        en: in std_logic;
        prime1, prime2: buffer integer
    );
end entity RandomPrimeGenerator;

architecture rtl of RandomPrimeGenerator is


begin
    process(clk)
        variable seed1: integer := 1;
        variable seed2: integer := 2;
        variable rand_int1, rand_int2: integer;
        variable is_prime1, is_prime2: boolean;

        --Random Number Generator Function
        impure function rand_int_gen(
            min: integer;
            max: integer
        ) return integer is
            variable randomValue: real;
        begin
            uniform(seed1, seed2, randomValue);
            return integer(round(randomValue * real(max - min + 1) +
real(min) - 0.5));
        end function;

    begin
        if rising_edge(clk) AND en = '1' then
            is_prime1 := false;
            is_prime2 := false;

            while not (is_prime1 and is_prime2) loop
                rand_int1 := rand_int_gen(0, 8);
                rand_int2 := rand_int_gen(0, 8);
```

```
                --Primality testing using trial division method
                -- Check if the first random number is prime
                is_prime1 := true;
                for i in 2 to integer(sqrt(real(rand_int1))) loop
                    if rand_int1 mod i = 0 then
                        is_prime1 := false;
                        exit;
                    end if;
                end loop;

                -- Check if the second random number is prime
                is_prime2 := true;
                for i in 2 to integer(sqrt(real(rand_int2))) loop
                    if rand_int2 mod i = 0 then
                        is_prime2 := false;
                        exit;
                    end if;
                end loop;
            end loop;

            -- Assign the final results
            prime1 <= rand_int1;
            prime2 <= rand_int2;
        end if;
    end process;



end architecture rtl;
```

- KeyGenerator.vhd



Gambar 6. Output Program KeyGenerator.vhd

Source Code:

```
library IEEE;
```

```
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity KeyGenerator is
    port (
        clk: in std_logic;
        en: in std_logic;
        prime1_key, prime2_key: buffer integer;
        encryption_key, decryption_key, n_key: out integer
    );
end entity KeyGenerator;

architecture rtl of KeyGenerator is

    component RandomPrimeGenerator is
        port (
            clk: in std_logic;
            en: in std_logic;
            prime1, prime2: out integer
        );
    end component RandomPrimeGenerator;

    --Coprime Testing Function
    function IsCoprime(A, B: INTEGER) return BOOLEAN is
        variable Remainder: INTEGER;
        variable TempA, TempB: INTEGER;
    begin
        TempA := A;
        TempB := B;
        if TempA > TempB then
            TempA := B;
            TempB := A;
        end if;

        loop
            Remainder := TempB mod TempA;
            exit when Remainder = 0;
            TempB := TempA;
            TempA := Remainder;
        end loop;

        return TempA = 1;
    end IsCoprime;

begin
```

```
    RandomPrimeGenerator1: RandomPrimeGenerator port map (
        clk => clk,
        en => en,
        prime1 => prime1_key,
        prime2 => prime2_key
    );

    process(clk)
        variable N, T, e, d: integer := 0;
    begin
        if rising_edge(clk) AND en = '1' then
            N := prime1_key * prime2_key;
            T := (prime1_key - 1) * (prime2_key - 1);

            --Algorithm to find e (encryption key)
            for i in 2 to T - 1 loop
                if T mod i /= 0 and N mod i /= 0 and IsCoprime(i, T) and
IsCoprime(i, N) then
                    e := i;
                    exit;
                end if;
            end loop;

            --Algorithm to find d (decryption key)
            for i in 1 to T loop
                if (e * i) mod T = 1 then
                    d := i;
                    exit;
                end if;
            end loop;
        end if;

        encryption_key <= e;
        decryption_key <= d;
        n_key <= N;
    end process;

    -- Additional logic for key generation (e.g., calculation of private
key 'd')
    -- e * d mod T = 1
    -- e must be < T and coprime with N and T
    -- Add your key generation logic here

end architecture rtl;
```

3.2. Analisa

- Sender.vhd

  Prime1_receiver dan prime2_receiver adalah bilangan prima yang dihasilkan oleh RandomPrimeGenerator. outCurr merupakan karakter yang sedang melalui proses enkripsi dan dekripsi. outputEncrypt adalah hasil enkripsi dari input, sedangkan outputDecrypt adalah hasil dekripsi. input_tb adalah input dari testbench, yang nantinya akan dikirim dari sender ke input receiver.

- Receiver.vhd

  E merupakan kunci enkripsi, d merupakan kunci dekripsi, dan n adalah kunci n. Jika tidak menggunakan testbench, input dapat dimasukkan secara manual, dan pada dasarnya tidak perlu ada nilai apapun di input receiver. Urutan statenya adalah sebagai berikut:
  FETCH -> ENCRYPT -> SHOW_ENCRYPTION -> DECRYPT -> SHOW_DECRYPTION.

- RandomPrimeGenerator.vhd

  Fungsi dan nilai e yang digunakan dalam enkripsi sama seperti yang dihasilkan oleh KeyGenerator. Di sini, meskipun generator angka acak dapat digunakan, di penerima akan selalu menggunakan prime1 dan prime2 yang memiliki nilai 3 dan 7. Mengapa? Karena jika setiap huruf menggunakan prime1 dan prime2 yang berbeda, maka kunci enkripsi (e key), kunci dekripsi (d key), dan kunci n (n key) akan berbeda untuk setiap huruf. Hal ini menyebabkan proses enkripsi-dekripsi menjadi salah. Jadi, dapat dikatakan bahwa urutan angka acak yang digunakan pada penerima hanya menggunakan yang pertama kali muncul. Ini tidak benar-benar acak di penerima, tetapi informasi ini dimasukkan ke dalam laporan untuk menunjukkan bahwa RandomPrimeGenerator berfungsi. Pada bagian pengembangan atau catatan kesalahan program, akan dijelaskan bahwa penerima tidak dapat mengambil nilai prima selain prime1 dan prime2 yang pertama.

- KeyGenerator.vhd

  En digunakan untuk mengaktifkan komponen KeyGenerator sehingga dapat berjalan. Nilai en diperoleh dari receiver dan fungsinya adalah agar KeyGenerator tidak terus-menerus menghasilkan kunci saat proses enkripsi-dekripsi berlangsung di receiver. Jika proses berjalan terus menerus, maka akan muncul kesalahan karena receiver akan mendapatkan kunci pertama, sedang diproses, dan sebelum proses selesai, masuk lagi dengan kunci kedua yang memiliki nilai berbeda. Oleh karena itu, en dinonaktifkan saat receiver sudah memiliki kunci, dan kondisi ini dapat dilihat pada bagian state FETCH di receiver.

BAB IV

KESIMPULAN

4.3. Kesimpulan

RSA Encryptor yang telah kelompok kami buat adalah program komputer yang dirancang untuk mengubah informasi menjadi bentuk yang sulit dimengerti atau diakses oleh orang yang tidak berwenang. Program ini menggunakan algoritma enkripsi RSA untuk melindungi data dari akses yang tidak sah. Dimana hanya beberapa orang yang memiliki kunci dekripsi yang benar sajalah yang dapat membaca pesan tersebut dengan benar.

Dengan begitu, program enkripsi RSA dapat membantu menjaga kerahasiaan pesan atau data yang dikirimkan melalui jaringan atau disimpan di perangkat komputer. Hal tersebut memberikan lapisan perlindungan tambahan terhadap akses yang tidak sah dan membantu memastikan keamanan informasi.

Secara keseluruhan, laporan akhir praktikum ini menunjukkan bahwa RSA Encryption untuk surat-menyurat yang dibuat dalam bahasa VHDL telah berhasil mencapai tujuan yang diinginkan dan dapat diimplementasikan secara efektif dalam melakukan enkripsi data untuk melindungi kerahasiaan pesan dari satu pengguna ke pengguna lain.

4.4. Refleksi

Namun, sejatinya masih terdapat beberapa hal yang dapat dikembangkan lebih jauh lagi, seperti pada komponen Reciever.vhd yang hanya dapat membaca sekuens prime pertama saja. Apabila nilai dari prime berubah maka akan terjadi kesalahan pada program. Lalu, terdapat juga sebuah permasalahan ketika kami melakukan sintesis program kami di aplikasi Quartus. Kesalahan tersebut terjadi karena adanya sebuah keinkompatibilitasan pada program kami dengan *compiler* Quartus. Maka dari itu, kami menggantinya dengan sebuah block diagram yang dibuat menggunakan draw.io (https://app.diagrams.net/).

Kedepannya kami harap kami dapat lebih menyempurnakan beberapa kesalahan yang terdapat pada program kami ini.

REFERENSI

Margaret Rouse (2017) *What is a random number generator (RNG)? - definition from Techopedia*. Available at: https://www.techopedia.com/definition/9091/random-number-generator-rng (Accessed: 24 December 2023).

Cobb, M. (2021) *What is the RSA algorithm? definition from searchsecurity, Security*. Available at: https://www.techtarget.com/searchsecurity/definition/RSA (Accessed: 24 December 2023).

How does a random number generator work?: Encyclopedia (no date) *How does a Random Number Generator work? | Encyclopedia*. Available at: https://www.hypr.com/security-encyclopedia/random-number-generator (Accessed: 24 December 2023).

Jensen, J.J. (2023) *How to create a finite-state machine in VHDL, VHDLwhiz*. Available at: https://vhdlwhiz.com/finite-state-machine/ (Accessed: 24 December 2023).

Trial Division algorithm for prime factorization (2021) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/trial-division-algorithm-for-prime-factorization/ (Accessed: 24 December 2023).

What is RSA? how does an RSA work? (2020) *Encryption Consulting*. Available at: https://www.encryptionconsulting.com/education-center/what-is-rsa/ (Accessed: 24 December 2023).

RSA algorithm in cryptography (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/ (Accessed: 24 December 2023).