

Proyek

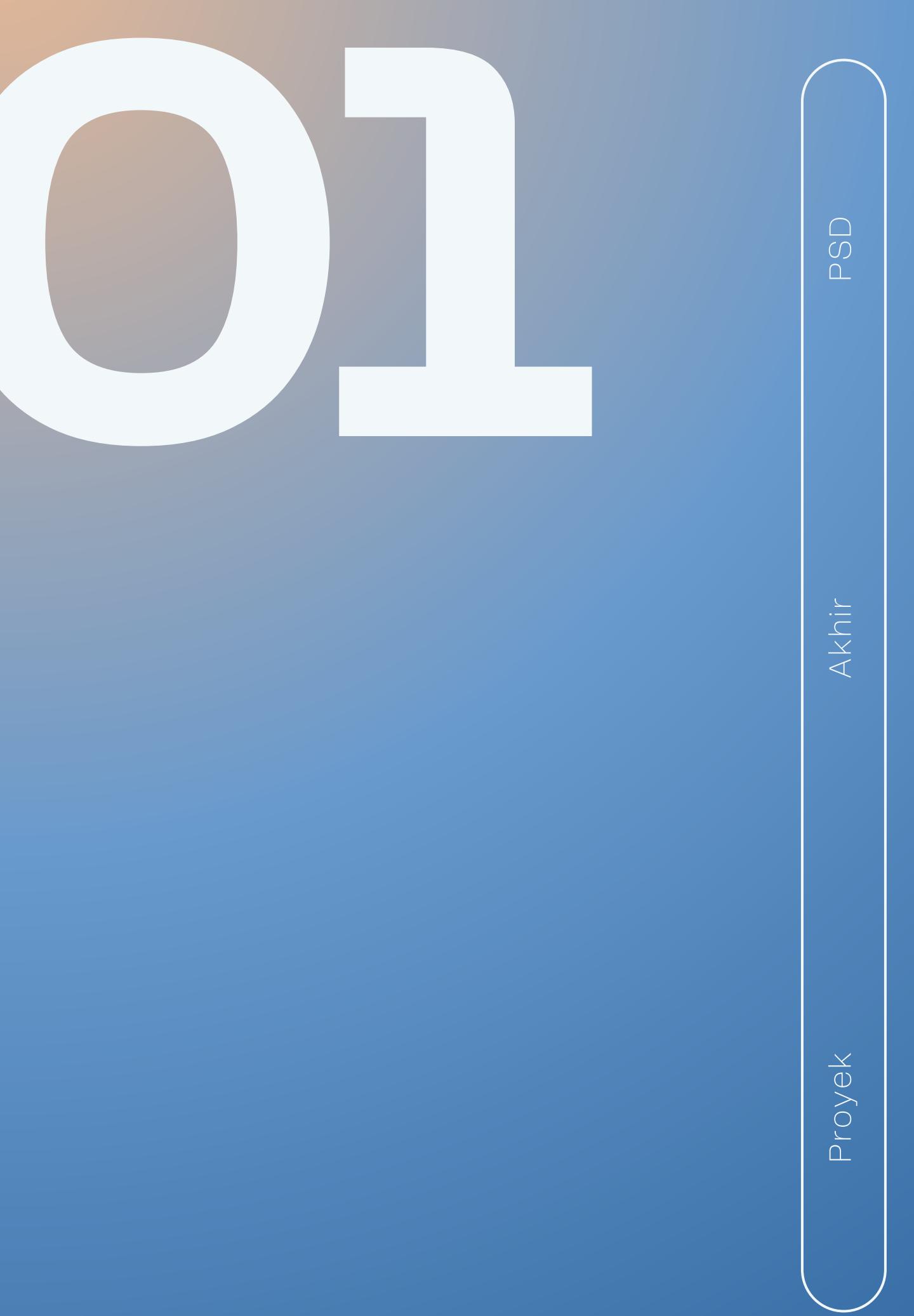
Akhir

PSD

FINAL PROJECT PSD

SISTEM ENKRIPSI RSA
DENGAN RANDOM PRIME
GENERATOR BERBASIS VHDL

Presented By: Kelompok 13



TUJUAN

- Mengimplementasikan pemrograman menggunakan bahasa VHDL
- Merancang metode enkripsi RSA yang akan mengenkripsi pesan berbentuk string
- Memenuhi nilai praktikum Perancangan Sistem Digital

LATAR BELAKANG

Keamanan data sangat penting dalam komunikasi digital, karena data sering kali rentan terhadap serangan di jaringan yang tidak aman. RSA (Rivest-Shamir-Adleman) adalah algoritma kriptografi yang menggunakan kunci publik untuk enkripsi dan kunci privat untuk dekripsi, dengan keamanan berdasarkan faktorisasi bilangan besar. Meskipun aman, RSA dalam perangkat lunak cenderung lambat.

Implementasi di perangkat keras dengan VHDL dapat meningkatkan kecepatan pemrosesan dan keamanan. Proyek ini bertujuan mengimplementasikan RSA menggunakan VHDL dengan generator bilangan prima acak untuk menghasilkan kunci publik dan privat serta proses enkripsi-dekripsi yang cepat dan aman.

CARA KERJA

RSA encryptor bekerja sebagai berikut:

Entitas Sender bertindak sebagai pengirim pesan. Pada tahap awal, Sender mengirimkan pesan ke entitas Receiver untuk dienkripsi dan didekripsi menggunakan algoritma RSA. Ini dilakukan dengan menggunakan komponen Receiver sebagai komponen utama yang berinteraksi dengan entitas lain untuk melakukan enkripsi dan dekripsi pesan.

Entitas Receiver sebagai komponen utama menerima pesan dari Sender, lalu melakukan enkripsi dan dekripsi menggunakan kunci publik dan kunci privat yang dihasilkan oleh entitas KeyGenerator. Proses enkripsi dilakukan dengan mengubah karakter pesan menjadi bilangan bulat, kemudian dilakukan operasi eksponensial dan modulo dengan kunci publik. Pesan terenkripsi dikirimkan kembali ke Sender, dan proses dekripsi dilakukan di Receiver dengan menggunakan kunci privat yang sesuai.



CARA KERJA

Entitas KeyGen berfungsi untuk menghasilkan kunci publik dan kunci privat yang digunakan dalam proses enkripsi dan dekripsi pesan. Komponen PrimeGen pada KeyGen menghasilkan bilangan prima secara acak yang menjadi dasar untuk pembuatan kunci publik dan kunci privat dalam algoritma RSA.

Proses algoritma RSA dijalankan dalam entitas Receiver. Ketika pesan diterima, karakter pesan diubah menjadi bilangan bulat, kemudian dilakukan operasi matematika sesuai langkah-langkah enkripsi RSA. Hasil enkripsi dikirim kembali ke Sender untuk ditampilkan, sementara dekripsi dilakukan di Receiver menggunakan kunci privat yang sesuai, dan hasil dekripsi ditampilkan.



STATE DIAGRAM

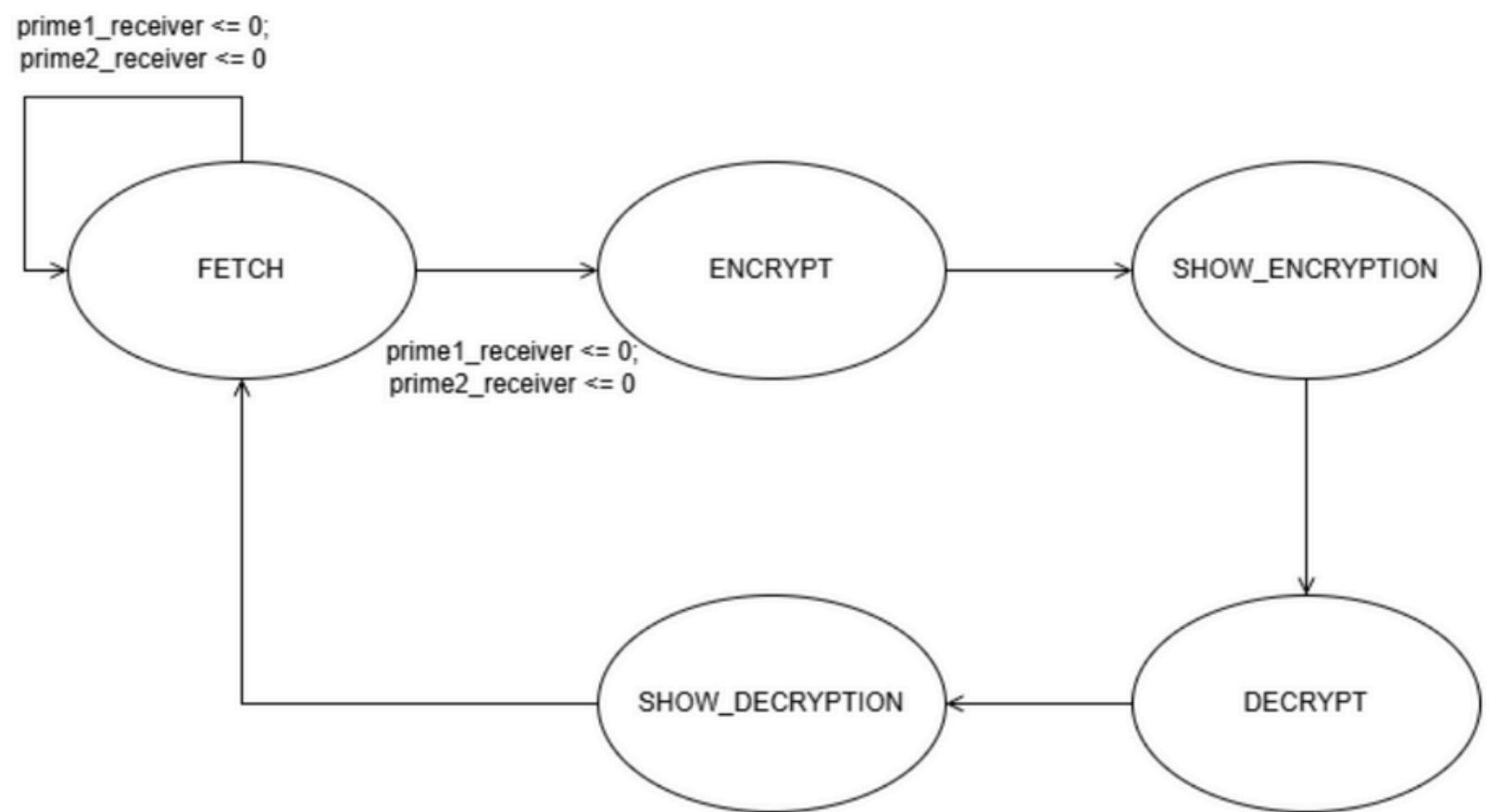
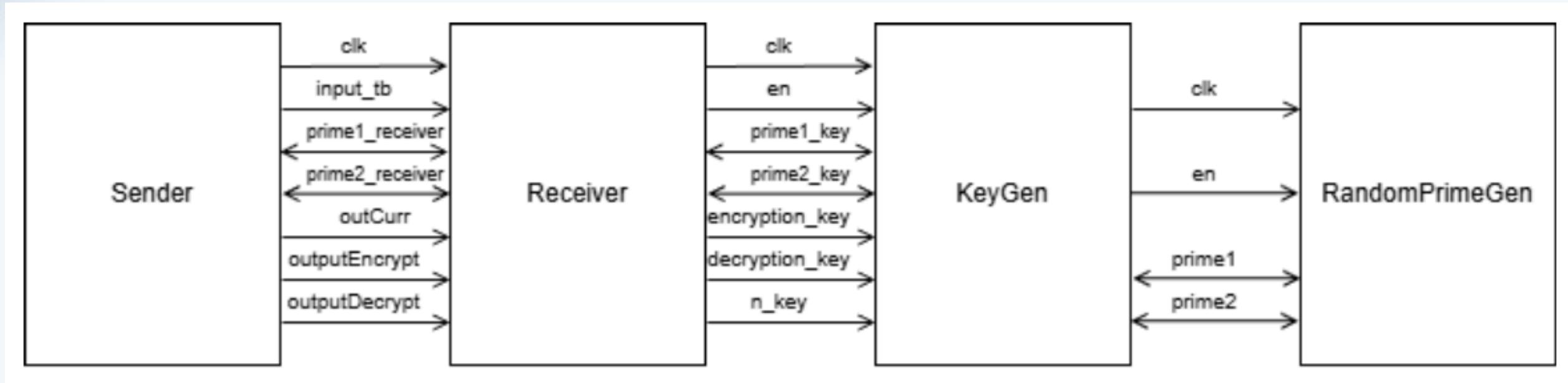


Diagram ini menunjukkan siklus lengkap dari pengambilan data, enkripsi, dan dekripsi dengan kondisi sebagai berikut:

- **FETCH**: Menunggu input valid (`Prime1 & Prime2 > 0`). Jika tidak valid, sistem tetap standby.
- **ENCRYPT**: Memproses enkripsi pada data yang valid.
- **SHOW_ENCRYPTION**: Menampilkan hasil data terenkripsi (ciphertext).
- **DECRYPT**: Mengembalikan data ke bentuk aslinya (plaintext).
- **SHOW_DECRIPTION**: Menampilkan hasil pemulihan data.
- **Return**: Sistem kembali ke **FETCH** untuk memulai siklus baru.

BLOCK DIAGRAM



- Sender (Testbench): Berfungsi sebagai lingkungan pengujian yang menyuplai sinyal clock dan data input ke dalam sistem, sekaligus memonitor output yang dihasilkan, baik berupa data terenkripsi maupun hasil dekripsi.
- Receiver: Unit pemrosesan utama yang bertanggung jawab mengeksekusi algoritma enkripsi dan dekripsi menggunakan kunci yang telah disediakan.
- KeyGen: Modul generator yang bertugas mengkomputasi dan menghasilkan parameter kriptografi, meliputi kunci enkripsi, kunci dekripsi, serta nilai modulus (n_key) berdasarkan input bilangan prima.
- RandomPrimeGen: Pembangkit bilangan acak yang secara spesifik menyediakan dua bilangan prima (prime1 dan prime2) sebagai fondasi matematis untuk pembuatan kunci.

Testbench

- Deklarasi Entity Sender: Bagian ini mendefinisikan port untuk sinyal clock (clk), dua bilangan prima (prime1_receiver dan prime2_receiver), serta output karakter yang digunakan dalam proses enkripsi dan dekripsi (outCurr, outputEncrypt, outputDecrypt).
- Instansiasi Komponen Receiver: Pada bagian ini, komponen Receiver dihubungkan ke port-port yang sudah dibuat di Sender. Receiver berfungsi untuk melakukan proses enkripsi dan dekripsi berdasarkan data yang masuk.
- Input untuk Testbench: Kode ini juga berperan sebagai sumber data (pengirim). Variabel input_tb berisi sebuah string yang nantinya dikirim ke Receiver untuk diproses.
- Pengujian melalui Testbench: Testbench menyediakan sinyal input yang akan diolah oleh Receiver, lalu mengamati hasil keluaran dari proses enkripsi dan dekripsi untuk memastikan bahwa komponen bekerja dengan benar.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity Sender is
6      port (
7          clk: in std_logic;
8          prime1_receiver, prime2_receiver: buffer integer;
9          outCurr: out character;
10         outputEncrypt: out character;
11         outputDecrypt: out character
12     );
13 end entity Sender;
14
15 architecture rtl of Sender is
16     component Receiver is
17         port (
18             clk: in std_logic;
19             input: in string (1 to 19);
20             prime1_receiver, prime2_receiver: buffer integer;
21             outCurr: out character;
22             outputEncrypt: out character;
23             outputDecrypt: out character
24         );
25     end component Receiver;
26
27     signal input_tb: string(1 to 19) := "B C D E F G H I J K";
28 begin
29     Receiver1: Receiver port map (
30         clk => clk,
31         input => input_tb,
32         prime1_receiver => prime1_receiver,
33         prime2_receiver => prime2_receiver,
34         outCurr => outCurr,
35         outputEncrypt => outputEncrypt,
36         outputDecrypt => outputDecrypt
37     );
38 end architecture rtl;
```

Entity Sender

```
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;
```

Entity Sender digunakan untuk mendefinisikan interface yang dipakai pada testbench.

- clk: Sinyal clock yang berfungsi sebagai sumber sinkronisasi.
- prime1_receiver, prime2_receiver: Tipe buffer integer yang menyimpan dua bilangan prima yang dikirimkan ke komponen lainnya.
- outCurr: Karakter yang sedang diproses (karakter input).
- outputEncrypt: Karakter hasil proses enkripsi.
- outputDecrypt: Karakter hasil proses dekripsi.

Tipe BUFFER digunakan ketika suatu sinyal harus bisa dibaca sekaligus ditulis di dalam testbench. Pada konteks ini, nilai bilangan prima dapat diubah oleh komponen lain lalu dibaca kembali oleh sistem.

Architecture rtl

```
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;
```

Bagian architecture rtl mendefinisikan komponen-komponen yang digunakan di dalam testbench. Pada bagian ini, komponen yang diuji adalah Receiver.

- Komponen Receiver memiliki port berupa sinyal clock, input berupa string, dua bilangan prima sebagai masukan, serta keluaran yang menunjukkan karakter yang sedang diproses, hasil enkripsi, dan hasil dekripsi.

Mapping

```
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;
```

- Sinyal input_tb: Sinyal ini berisi sebuah string yang memuat karakter-karakter yang akan diproses oleh sistem, baik untuk enkripsi maupun dekripsi. Nilai string tersebut menjadi data utama yang akan dibaca oleh Receiver.
- Receiver1: Bagian port map ini menghubungkan komponen Receiver dengan seluruh sinyal yang berasal dari Sender. Pada bagian ini, input berupa string (input_tb), dua bilangan prima (prime1_receiver dan prime2_receiver), serta keluaran berupa karakter hasil pemrosesan (outCurr, outputEncrypt, outputDecrypt) semuanya dipetakan ke Receiver.

- Testbench ini berfungsi untuk menguji komponen Receiver yang menangani proses enkripsi dan dekripsi berdasarkan karakter-karakter dari input_tb.
- Sinyal prime1_receiver dan prime2_receiver berisi dua bilangan prima yang digunakan sebagai dasar perhitungan enkripsi dan dekripsi.
- Receiver akan memproses karakter input satu per satu, melakukan enkripsi dan dekripsi, lalu mengirimkan hasilnya ke outputEncrypt dan outputDecrypt.

Testbench ini mensimulasikan proses enkripsi dan dekripsi untuk string "B C D E F G H I J K", menggunakan bilangan prima yang dikirimkan melalui prime1_receiver dan prime2_receiver. Tujuan dari kode ini adalah memastikan bahwa implementasi RSA berjalan dengan benar, dengan memproduksi hasil enkripsi dan dekripsi sesuai dengan yang diharapkan.

Receiver

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 entity Receiver is
6   port (
7     clk: in std_logic;
8     input: in string (1 to 19) := "B C D E F G H I J K";
9     prime1_receiver, prime2_receiver: buffer integer;
10    outCurr: out character;
11    outputEncrypt: out character;
12    outputDecrypt: out character
13  );
14 end entity Receiver;
15
16 architecture rtl of Receiver is
17
18   component KeyGenerator is
19     port (
20       clk: in std_logic;
21       en: in std_logic;
22       prime1_key, prime2_key: buffer integer;
23       encryption_key, decryption_key, n_key: out integer
24     );
25   end component KeyGenerator;
26
27   --STATES
28   type state is (FETCH, ENCRYPT, SHOW_ENCRYPTION, DECRYPT, SHOW_DECRYPTION);
29
30   --SIGNAL FOR STATES
31   signal currstate, nextstate: state;
32
33   --ENABLE SIGNAL
34   signal en_receiver: std_logic := '1';
35   signal e, d, n: integer;
36
37 begin
38   KeyGen: KeyGenerator port map (
39     clk => clk,
40     en => en_receiver,
41     prime1_key => prime1_receiver,
42     prime2_key => prime2_receiver,
43     encryption_key => e,
44     decryption_key => d,
45     n_key => n
46   );

```

```

48   process(clk)
49     --TEMPORARY VARIABLES TO STORE THE INTEGER OF A CHAR
50     variable i_int, e_int, d_int: integer := 0;
51     variable pc: integer := 1;
52   begin
53
54     if rising_edge(clk) then
55       outCurr <= input(pc);
56       case currstate is
57         when FETCH =>
58           i_int := character'pos(input(pc)) - 64;
59
60           if prime1_receiver > 0 AND prime2_receiver > 0 then
61             en_receiver <= '0';
62             nextstate <= ENCRYPT;
63           end if;
64         when ENCRYPT =>
65           e_int := (i_int ** e) mod n;
66           nextstate <= SHOW_ENCRYPTION;
67         when SHOW_ENCRYPTION =>
68           outputEncrypt <= character'val(e_int + 64);
69           nextstate <= DECRYPT;
70         when DECRYPT =>
71           d_int := (e_int ** d) mod n;
72           nextstate <= SHOW_DECRYPTION;
73         when SHOW_DECRYPTION =>
74           outputDecrypt <= character'val(d_int + 64);
75           pc := pc + 1;
76           nextstate <= FETCH;
77       end case;
78     end if;
79   end process;
80
81   process(clk)
82   begin
83     if rising_edge(clk) then
84       currstate <= nextstate;
85     end if;
86   end process;
87
88
89 end architecture rtl;

```

- Inisialisasi Input: Sistem membaca setiap karakter dalam string input secara berurutan lalu mengubahnya menjadi nilai numerik.
- Proses Enkripsi: Nilai numerik dari karakter tersebut dienkripsi menggunakan rumus $(i_{int}^{**} e) \bmod n$.
- Proses Dekripsi: Nilai terenkripsi kemudian dikembalikan ke bentuk semula melalui perhitungan $(e_{int}^{**} d) \bmod n$.
- Menampilkan Hasil: Nilai yang telah didekripsi diubah kembali menjadi karakter dan disimpan pada outputDecrypt.
- Pengulangan: Langkah-langkah ini dijalankan berurutan untuk setiap karakter dalam string sampai seluruh data selesai diproses.

Deklarasi State Machine

```
--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;
```

Bagian ini menentukan jenis-jenis state (seperti FETCH, ENCRYPT, dan lainnya) yang nantinya digunakan untuk mengatur jalannya proses enkripsi maupun dekripsi.

Proses Transisi State

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

Kode ini mengatur perpindahan dari satu state ke state berikutnya setiap kali clock mendeteksi rising edge, sesuai dengan hasil pemrosesan sebelumnya.

Proses Enkripsi dan Dekripsi

```
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;
```

Bagian ini menangani alur utama enkripsi dan dekripsi, yaitu memproses setiap karakter secara berurutan menggunakan logika yang telah ditentukan untuk mengubah input menjadi output terenkripsi dan terdekripsi.

Conclusion : Kode ini merupakan bagian inti yang menjalankan modul enkripsi dan dekripsi berbasis RSA. Prosesnya meliputi mengubah karakter-karakter pada string input menjadi bentuk terenkripsi dan kemudian mendekripsinya kembali menggunakan kunci yang dibentuk dari dua bilangan prima.



RandomPrimeGen

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

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

architecture rtl of RandomPrimeGen 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;
            end if;
        end process;

        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;

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

- Pembuatan Angka Acak: Sistem membangkitkan nilai acak menggunakan fungsi rand_int_gen.
- Pemeriksaan Keprimaan: Setiap angka acak yang dihasilkan kemudian diuji apakah merupakan bilangan prima menggunakan metode trial division.
- Pengulangan: Jika angka yang diuji tidak memenuhi syarat sebagai bilangan prima, proses pengacakan dan pengecekan akan dilakukan kembali.
- Pemilihan Dua Bilangan Prima: Setelah dua bilangan prima yang valid ditemukan, keduanya diteruskan ke modul berikutnya untuk digunakan dalam proses pembentukan kunci RSA.

Inisialisasi Variabel

```
architecture rtl of RandomPrimeGen is

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

- seed1 dan seed2 digunakan sebagai nilai awal untuk menjalankan generator angka acak.
- rand_int1 dan rand_int2 menyimpan angka acak yang nantinya akan diuji apakah termasuk bilangan prima.
- is_prime1 dan is_prime2 berfungsi sebagai penanda yang menentukan apakah angka tersebut lolos sebagai bilangan prima.

Fungsi Generator Angka Acak

```
--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;
```

Fungsi rand_int_gen menghasilkan angka acak dalam rentang nilai min hingga max, dengan memanfaatkan metode uniform random number generation untuk mendapatkan nilai acak yang merata.

PSD

Akhir

Proyek

Pengecekan Prima

```
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;
```

Bagian ini memeriksa apakah rand_int1 dapat dibagi oleh angka lain selain 1 dan dirinya sendiri. Jika ditemukan pembagi yang valid, maka nilai is_prime1 akan diubah menjadi false, menandakan bahwa angka tersebut bukan bilangan prima.

Proses Pencarian Bilangan Prima:

- Tahap pertama adalah menghasilkan dua angka acak (rand_int1 dan rand_int2).
- Setelah itu, masing-masing angka diuji keprimaannya menggunakan pengecekan yang telah ditetapkan.
- Jika kedua angka dinyatakan prima, proses dihentikan dan angka tersebut diteruskan sebagai bilangan prima yang akan digunakan pada pembuatan kunci RSA.

```

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

entity KeyGen 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 KeyGen;

architecture rtl of KeyGen is

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

--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
    RandomPrimeGen1: RandomPrimeGen 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;
end architecture rtl;

```

KeyGen

Proyek Akhir

PSD

- Pengambilan Dua Bilangan Prima: Sistem mengambil dua bilangan prima yang dihasilkan oleh modul RandomPrimeGen sebagai dasar perhitungan kunci RSA.
- Proses Penentuan Kunci Enkripsi dan Dekripsi: Nilai kunci enkripsi (e) dan kunci dekripsi (d) dihitung menggunakan algoritma RSA, dengan memastikan bahwa keduanya memenuhi syarat coprime.
- Perhitungan Nilai N: Menghitung nilai N dari hasil perkalian kedua bilangan prima tersebut, yang kemudian digunakan sebagai bagian penting dalam proses enkripsi dan dekripsi data.

Perhitungan Nilai N dan T

```
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);
```

Proses ini mengkalkulasi nilai modulus (N) dan totient (T) menggunakan dua input bilangan prima yang diperoleh dari modul PrimeGen.

PSD

Akhir

Proyek

Fungsi Pencarian Nilai e (Kunci Enkripsi)

```
--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;
```

Menentukan nilai e yang bersifat saling prima (coprime) terhadap nilai T dan N dengan bantuan fungsi IsCoprime.

Pencarian Nilai d (Kunci Dekripsi)

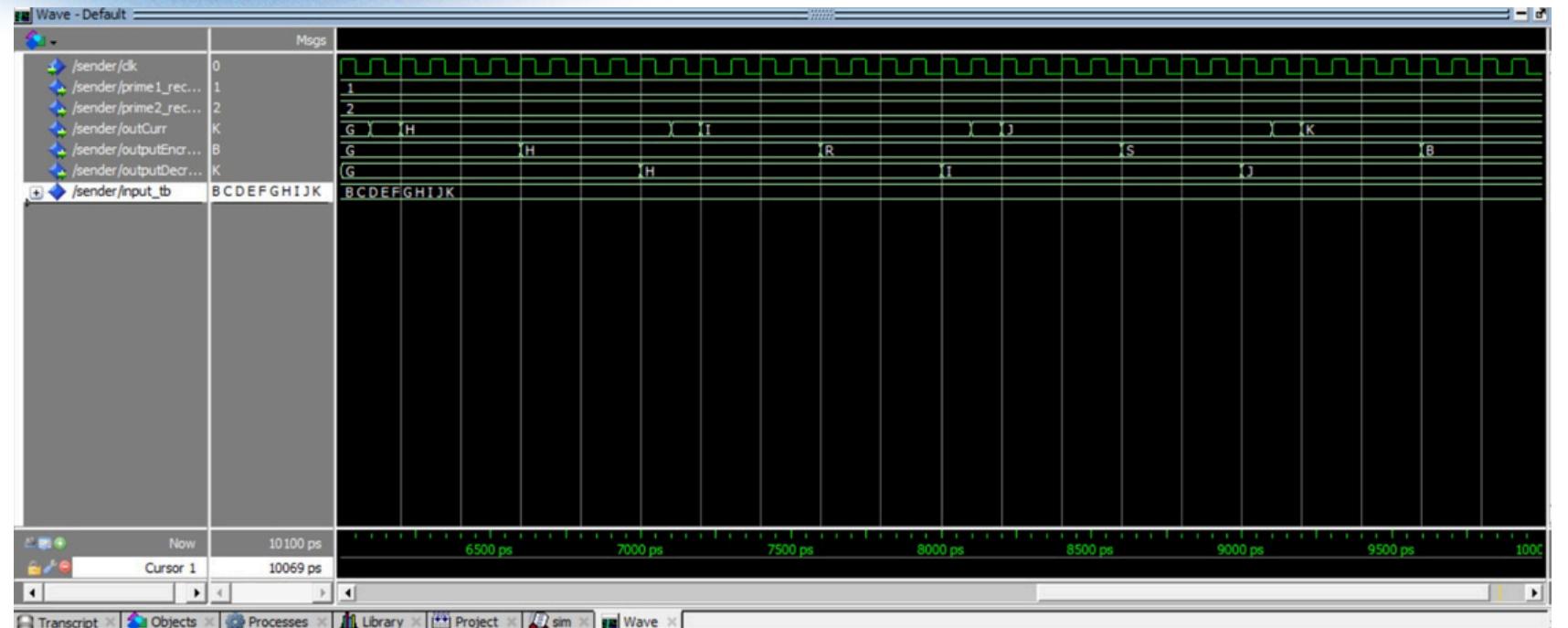
```
--Algorithm to find d (decryption key)
for i in 1 to T loop
    if (e * i) mod T = 1 then
        d := i;
        exit;
    end if;
end loop;
end if;
```

Memastikan bahwa nilai e berstatus coprime terhadap T dan N melalui fungsi IsCoprime guna menjamin tidak adanya faktor pembagi yang serupa.

Kesimpulan:

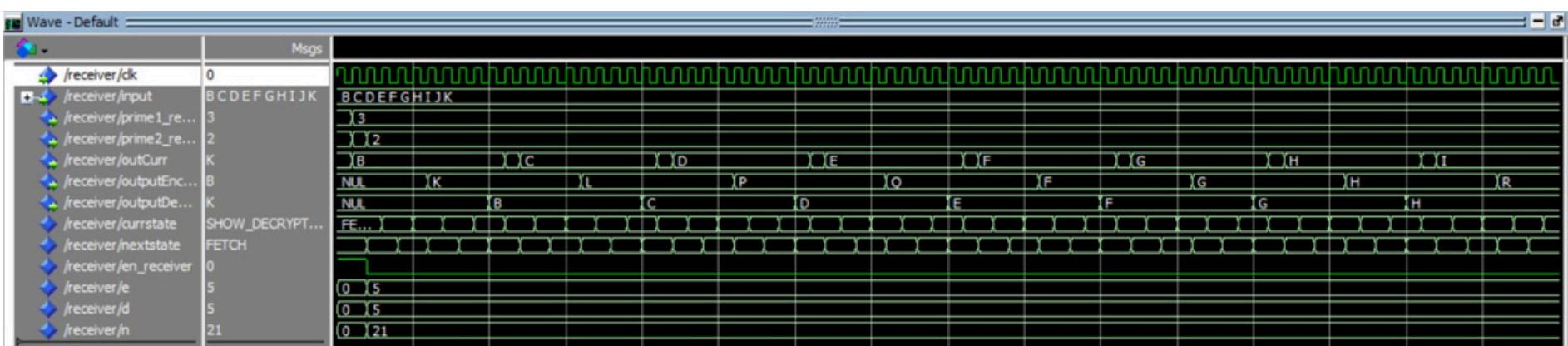
Kode ini bertujuan mengimplementasikan pembangkitan kunci RSA, baik untuk enkripsi maupun dekripsi. Tahapannya meliputi kalkulasi modulus (N) dan totient (T), penentuan kunci publik (e) yang relatif prima terhadap T dan N, serta komputasi kunci privat (d) berbasis aritmatika modular.

TESTING



Waveform (Testbench)

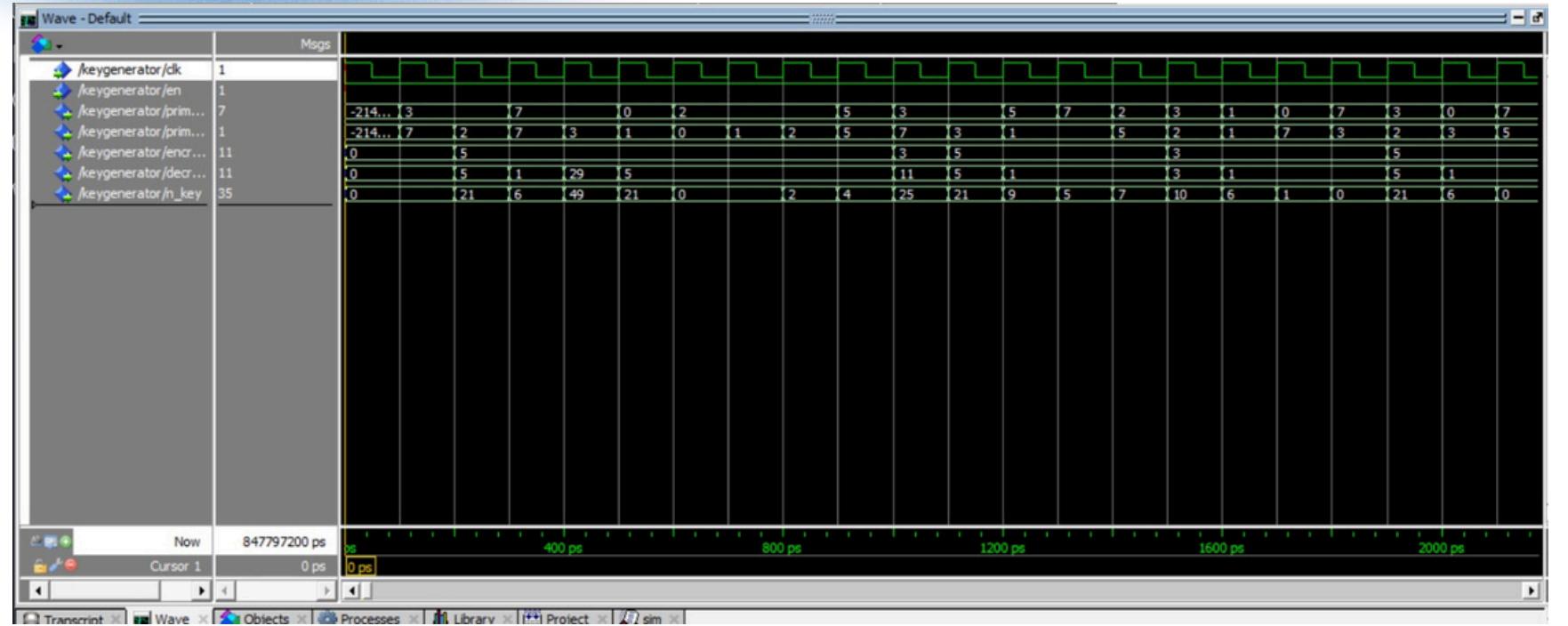
1. Input (A, B): Di sini kita lihat sinyal Input A dan B yang diubah-ubah untuk mengetes logikanya.
2. Output (Result): Output langsung berubah mengikuti inputnya, jadi logikanya udah jalan.
3. Validasi: Karena outputnya konsisten sama input, berarti modul ini sudah berjalan sesuai yang kita mau.



Waveform (Receiver)

1. Input (input, en_receiver): Bagian input ini akan menunjukkan kalau ada data masuk dan penerimanya lagi aktif
2. Sinyal Internal (prime1_receiver, prime2_receiver): Sinyal internal ini buat untuk memberi tahu status pengolahan datanya lagi udah sampai mana.
3. Output (outputDecrypt, outputEncrypt): Outputnya bakalan langsung ngasih liat hasil enkripsi atau dekripsinya.

TESTING



Waveform (KeyGen)

1. Input: Menggunakan clock dan enable untuk memproses dua input bilangan prima (prime1 & prime2).
2. Output: Menghasilkan set kunci lengkap: kunci enkripsi, kunci dekripsi, dan nilai modulus N.
3. Proses: Kalkulasi berjalan sinkron mengikuti clock ketika sinyal enable diaktifkan.



Waveform (RandomPrimeGen)

1. Clock (clk): Merepresentasikan sinyal referensi waktu yang berjalan secara periodik dan stabil untuk sinkronisasi sistem.
2. Enable (en): Berfungsi sebagai sinyal kendali (control signal) yang memicu aktivasi pemrosesan pada komponen lainnya.
3. Prime (prime1, prime2): Menampilkan luaran bilangan prima yang berubah secara dinamis pada setiap siklus eksekusi.

THANK YOU

KELOMPOK 13

- 1.Gede Rama Pradnya (2306161914)
- 2.Kelvin Ferrell Tjoe (2306205393)
- 3.Syifa Naila Maulidya (2406436940)
- 4.Ade Zaskia Azzahra (2406344353)