

EL 2002 Sistem Digital

Proyek Kelompok

Dokumen Pengembangan Produk

ChaChaChat

Kelompok 14

NIM	Nama
13223112	Ahmad Fatur Rohman
13223039	Dimas Anggiat
13223110	Julian Benedict
13223038	Kyla Asha Mulani
13223040	Mohammad Najmutsaqib

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

Riwayat Dokumen.....	2
Proposal.....	1
Latar Belakang.....	1
Spesifikasi Teknis.....	2
Rancangan.....	4
Konsep Sistem.....	4
Rancangan Sistem.....	6
ASM.....	6
Datapath.....	12
Control FSM.....	14
Implementasi dan Simulasi.....	18
Implementasi FPGA dan Pengujian.....	23
Lampiran.....	24

Riwayat Dokumen

Tabel 0.1 Riwayat Dokumen

Versi	Tanggal	Catatan Perubahan
v0.1	21 November 2024	Use case dan latar belakang proposal
v0.2	28 November 2024	Implementasi dan spesifikasi teknis proposal
v0.3	5 Desember 2024	Diagram rancangan awal dan penyelesaian proposal
v0.4	28 Desember 2024	ASM chart, FSM, dan konsep sistem akhir
v0.5	8 Januari 2025	Rancangan sistem final
v1.0	11 Januari 2025	Implementasi dan simulasi VHDL + FPGA
v1.1	12 Januari 2025	Implementasi dan simulasi VHDL + FPGA lengkap
v1.2	13 Januari 2025	Final

Proposal

Latar Belakang

Komunikasi menjadi sesuatu yang sangat penting sejak dulu hingga saat ini. Komunikasi bertujuan untuk menyampaikan pesan antara pengirim dengan penerima. Pesan dapat berupa tulisan, suara, gerakan dan lain sebagainya. Setiap jenis pesan ini akan berpengaruh pada bagaimana cara berkomunikasi antara pengirim dan penerima. Pesan yang berbeda akan membutuhkan cara komunikasi yang berbeda agar pesan dapat tersampaikan. Saat ini, dimana teknologi sudah serba maju dan modern, komunikasi menjadi lebih mudah untuk dilakukan. Apabila pada zaman dahulu ketika teknologi belum semaju sekarang, pesan berupa tulisan hanya dapat diantarkan melalui media fisika yang harus diantarkan langsung. Karena zaman saat ini sudah maju, maka cara mengirimkan pesan juga mengalami perkembangan yang lebih pesat. Seseorang dapat mengirimkan pesan (karakter) dari satu tempat ke tempat lain yang letaknya sangat berjauhan saat itu juga tanpa harus memerlukan waktu yang sangat lama. Semuanya menjadi lebih mudah dan cepat. Masalahnya adalah kecepatan dan kemudahan ini seringkali berbanding terbalik dengan keamanan, terutama keamanan data pada pesan yang dikirim.

Terlebih lagi tidak semua pesan yang dikirim boleh diketahui oleh banyak orang. Bagaimana jika pesan yang ingin dikirimkan adalah suatu pesan rahasia atau pribadi yang hanya boleh diketahui pengirim dan penerima? Oleh karena itu, pesan ini harus dikunci terlebih dahulu selama perjalanan baru dibuka kembali ketika sudah sampai pada penerima pesan yang diinginkan. Proses penguncian ini disebut enkripsi dan pengembalian pesan ke semula disebut sebagai dekripsi. Contoh ketika mengirim pesan teks pada aplikasi *Whatsapp*, maka pesan ini akan dienkripsi secara end-to-end sehingga hanya pengirim dan penerima yang mengetahuinya. Kali ini akan dilakukan pemodelan enkripsi ini dengan menggunakan algoritma Chacha20. Algoritma Chacha20 merupakan salah satu algoritma yang bertujuan untuk memberikan enkripsi yang cepat dan efisien. Pesan yang dimasukkan ke dalam algoritma ini, akan dienkripsi sehingga orang luar tidak dapat membaca pesan tersebut. Hanya orang yang memiliki kunci yang sama baru bisa membuka pesan dan mengetahui isinya.

Spesifikasi Teknis

Sistem yang akan dibuat pada kali ini akan menggunakan input berupa pesan yang akan dimasukkan, dan proses yang ingin dilakukan apakah mengenkripsi atau mendekripsi pesan yang ada. Jika seseorang ingin melakukan enkripsi, maka cukup memasukkan pesan yang ingin dienkripsi dan masuk ke mode enkripsi. Key dan nonce untuk enkripsi ini akan diproduksi oleh sistem dan akan diberitahukan kepada pengguna. Jika ingin mendekripsi, maka dapat dilakukan dengan memasukkan pesan terenkripsi serta key dan nonce pada sistem sehingga pesan asli dapat diketahui.

Tabel 1.1 Spesifikasi Sistem

No	Spesifikasi	Keterangan
1	Input plaintext	Merupakan pesan masukan yang akan dikirimkan dari pengirim kepada penerima. Pesan ini berupa karakter dalam bentuk string. Untuk panjang pesan dibatasi hingga 128 karakter karena rata-rata orang mengirim pesan di <i>whatsapp</i> tidak terlalu panjang.
2	Input Key	Key berfungsi untuk menghasilkan keystream bersama dengan input yang lain. Key ini memiliki panjang 32 byte atau 256 bit. Jika sistem sedang ingin melakukan enkripsi, maka key akan dihasilkan sendiri. Sedangkan pada proses dekripsi, key harus dimasukkan oleh pengguna.
3	Input Nonce	Nonce (<i>Number used once</i>) merupakan nilai yang hanya digunakan satu kali dalam proses enkripsi. Hal juga berfungsi untuk memastikan pada key yang sama, enkripsi akan menghasilkan output yang berbeda. Nonce memiliki panjang 12 byte atau 96 bit. Seperti key, nonce akan dihasilkan oleh sistem pada proses enkripsi. Jika ingin men-dekripsi pesan yang sudah terkunci, maka nonce akan dimasukkan manual oleh pengguna.
4	Mode	Sinyal penentu mode enkripsi atau dekripsi sistem (mode = '0' untuk enkripsi dan mode = '1' untuk dekripsi) yang ditentukan oleh button press.
5	Output Ciphertext	Ciphertext merupakan pesan yang sudah dienkripsi (dalam bentuk hex). Ciphertext ini berasal dari operasi XOR antara pesan masukan (plaintext) dengan KeyStream yang sudah dihasilkan sebelumnya.
6	Output 7 segment	7 segment pada FPGA sebagai penanda mode apa yang sedang dikerjakan oleh FPGA saat itu apakah enkripsi atau dekripsi untuk disesuaikan dengan input yang diberikan pengguna.

Pembagian Tugas

Tabel 1.2 Pembagian Tugas

NIM	Nama	Tugas
13223112	Ahmad Fatur Rohman	Komunikasi dari FPGA ke RealTerm, FSM, Integrasi
13223039	Dimas Anggiat	Neo-TRNG, mux n-bit
13223110	Julian Benedict	Membuat file txt menggunakan python, shift register nbit
13223038	Kyla Asha Mulani	Komunikasi dari RealTerm ke FPGA
13223040	Mohammad Najmutsaqib	Algoritma Chacha, Integrasi dengan UART

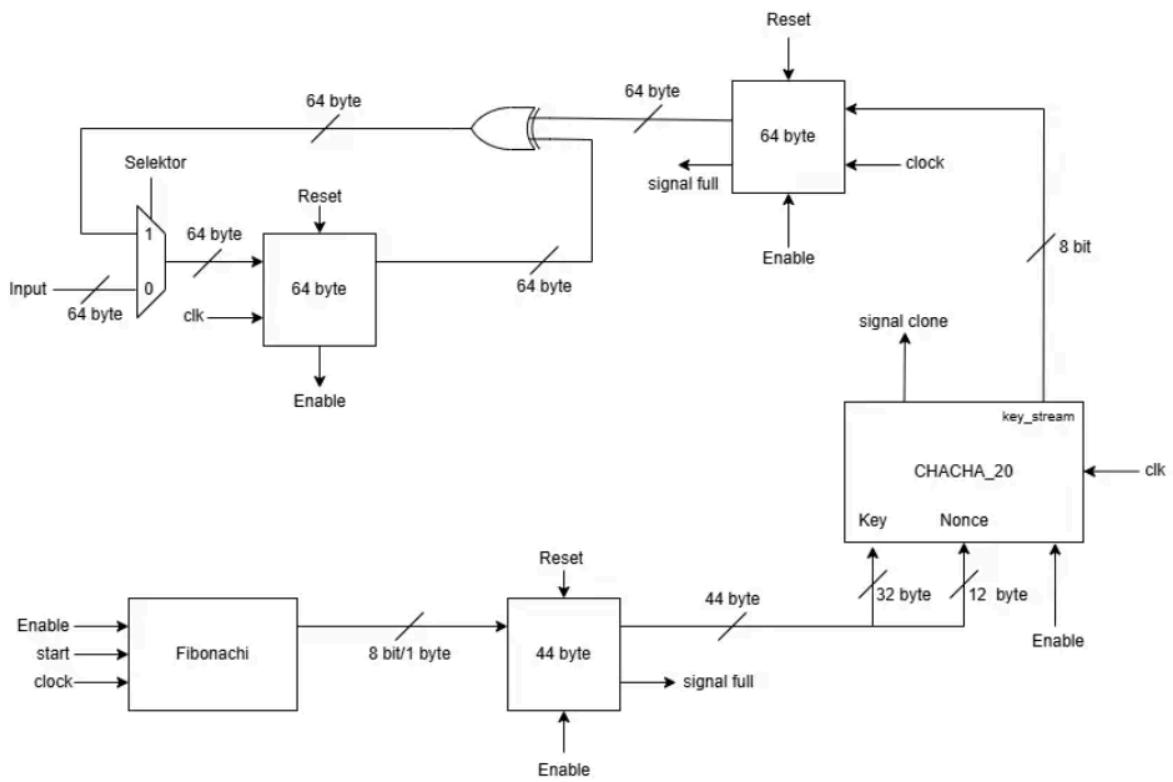
Rancangan

Konsep Sistem

Sistem kali ini menggunakan *software* pendukung berupa python. Jika ingin melakukan proses enkripsi, maka python akan meminta sebuah nama kontak untuk chatnya di enkripsi oleh ChaCha20, lalu sistem akan membuka sebuah chrome window yang membuka web whatsapp yang harus login menggunakan QR code, lalu Python akan mengambil chat terakhir dari nama kontak tersebut dan menyimpan chatnya dalam bentuk file **.txt**. Chat terakhir yang tersimpan dalam kontak inilah yang akan menjadi input text pada FPGA. Namun, jika ingin melakukan mode dekripsi, maka cukup memasukkan pesan terkunci pada terminal untuk mengetahui pesan awalnya.

Terdapat komponen kedua yaitu Fibonacci LFSR (Linear feedback shift register) akan menghasilkan key 256-bit dan nonce 96-bit ketika sedang proses enkripsi. Komponen ini akan menghasilkan output random sebesar 16 bit. Namun, jika mode yang akan digunakan adalah dekripsi, maka key dan nonce dimasukkan oleh pengguna bukan dari Fibonacci. Jika mode yang dipilih enkripsi, maka ketika sistem diaktifkan, akan dihasilkan keystream dari algoritma chacha20. Algoritma ini menggunakan key dan nonce yang berasal dari Fibonacci LFSR. Bersama dengan counter dan konstanta, akan dihasilkan *key stream* yang akan di XOR dengan input pesan tadi. Output yang akan diberikan adalah hasil enkripsi, key dan nonce yang digunakan dalam proses pembuatan *key stream*.

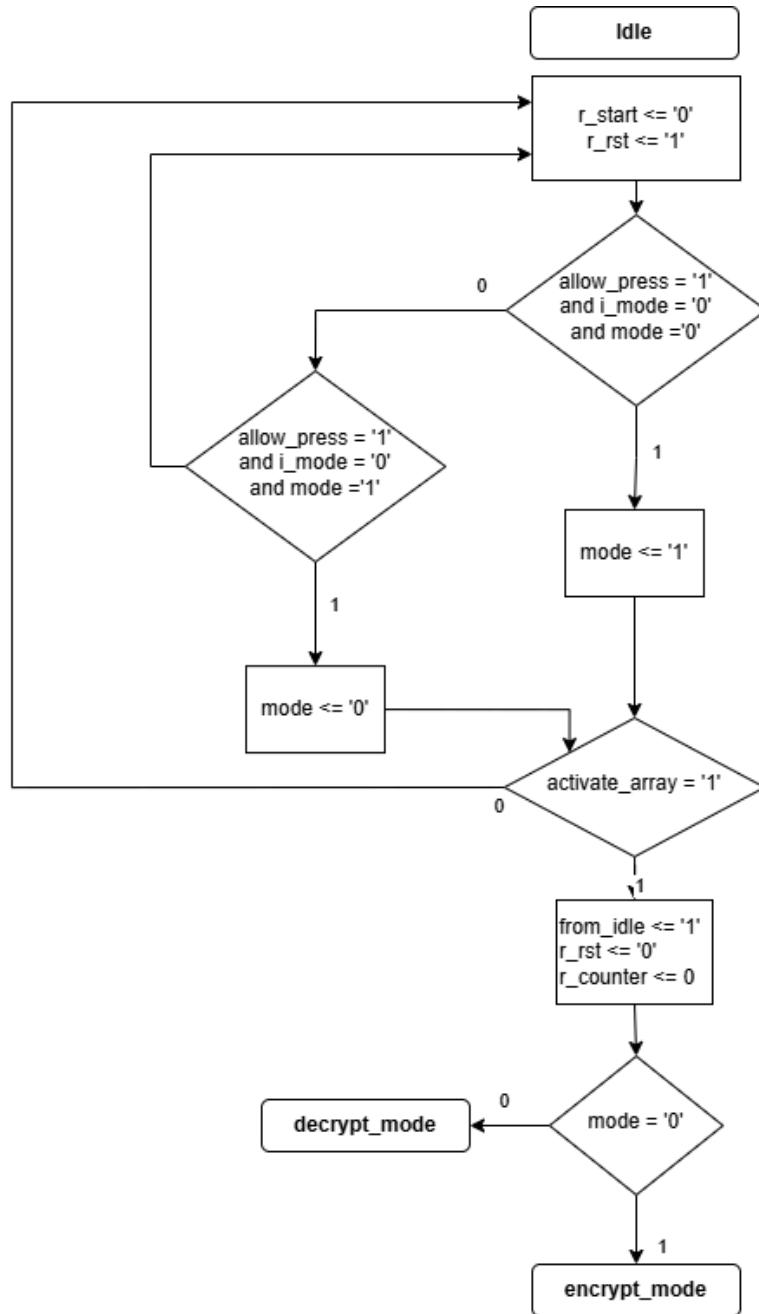
Pada mode dekripsi, pengguna akan diminta untuk memasukkan input text yang sudah terkunci (terenkripsi), serta key dan nonce yang sebelumnya digunakan untuk mengunci pesan. Karena, pengguna harus mengetahui key dan nonce yang tepat agar dapat membuka pesan terkunci tersebut. Output yang akan dihasilkan adalah pesan asli yang belum terkunci. Untuk prosesnya, sama seperti sebelumnya. Dengan key dan nonce yang dimasukkan akan dihasilkan *key stream* melalui algoritma chacha20. Nantinya *key stream* ini akan di XOR kan dengan pesan masukan.



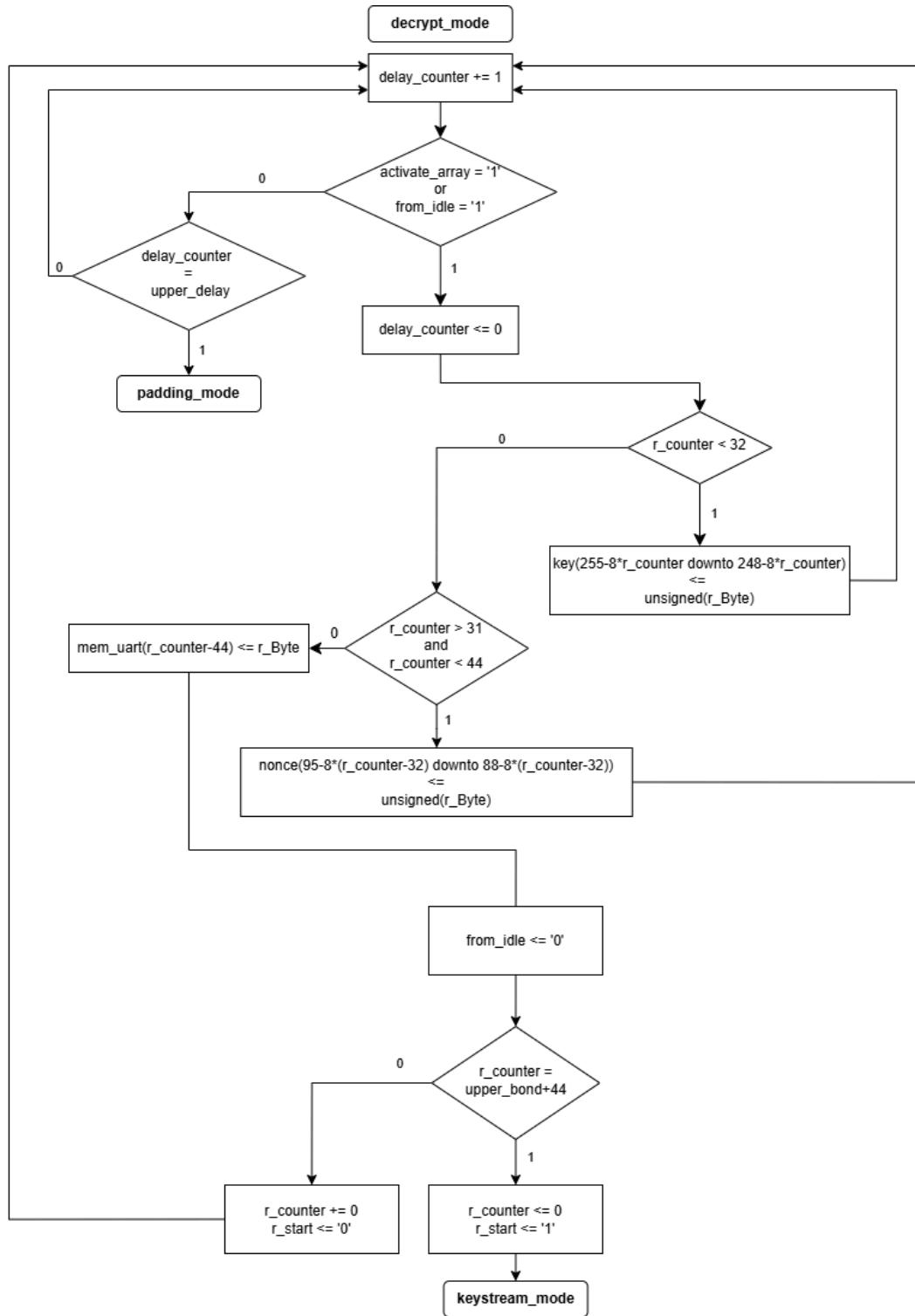
Gambar 2.1 Datapath rancangan awal

Rancangan Sistem

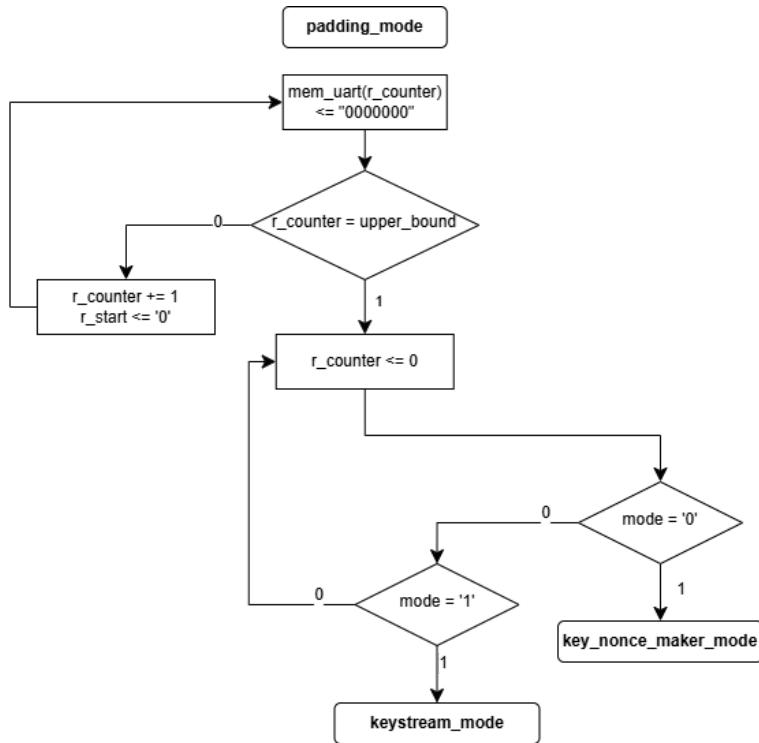
ASM



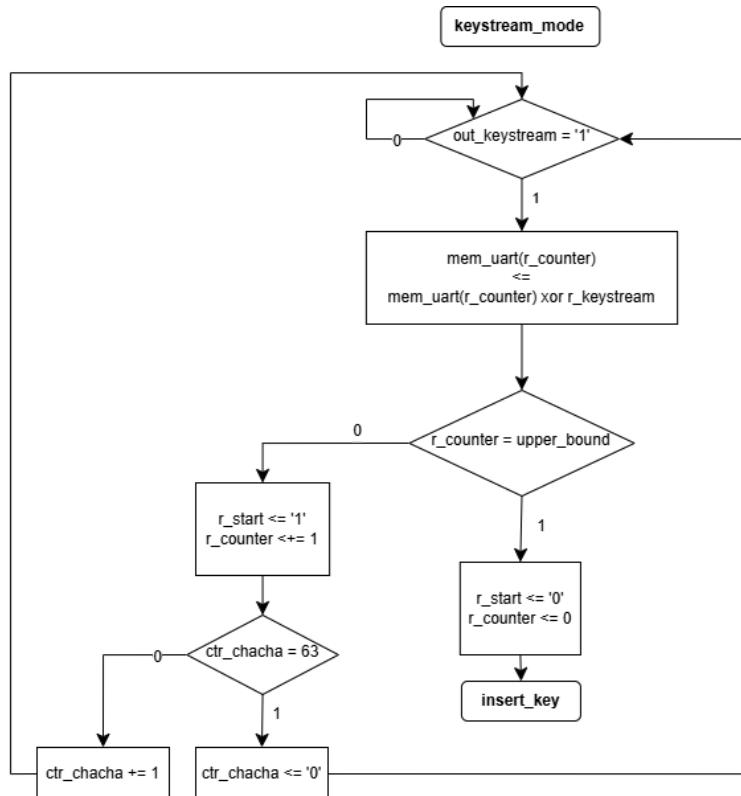
Gambar 3.1.1 ASM chart Idle state



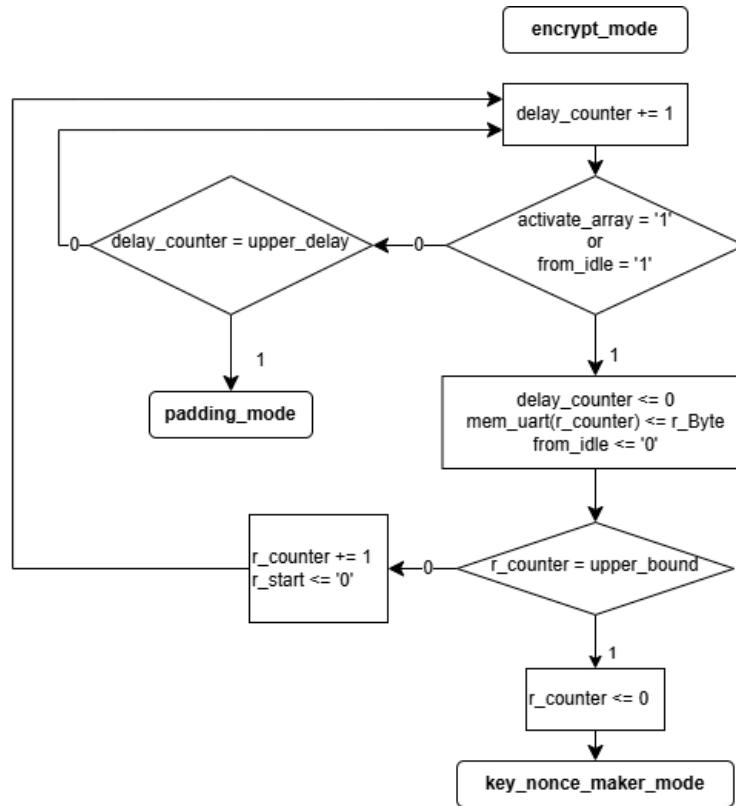
Gambar 3.1.2 ASM chart `decrypt_mode` state



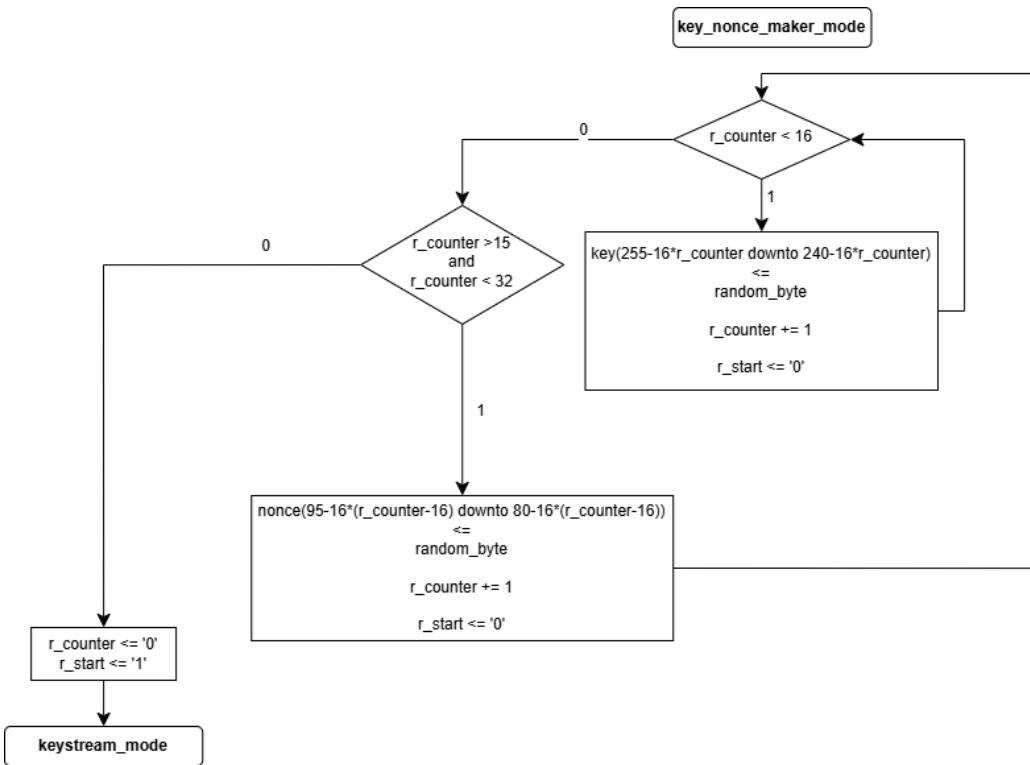
Gambar 3.1.3 ASM chart `padding_mode` state



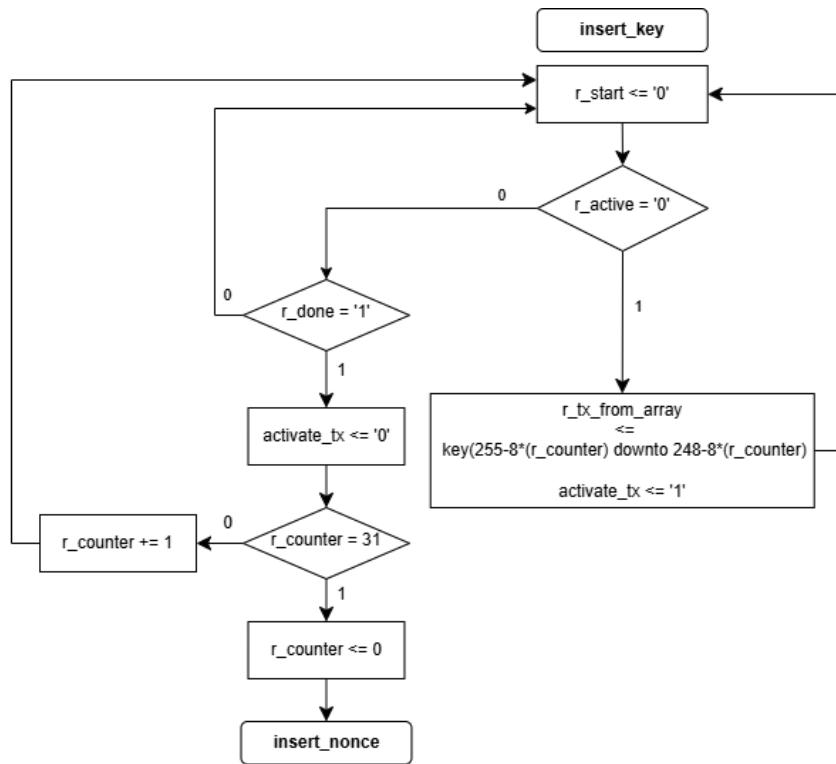
Gambar 3.1.4 ASM chart `keystream_mode` state



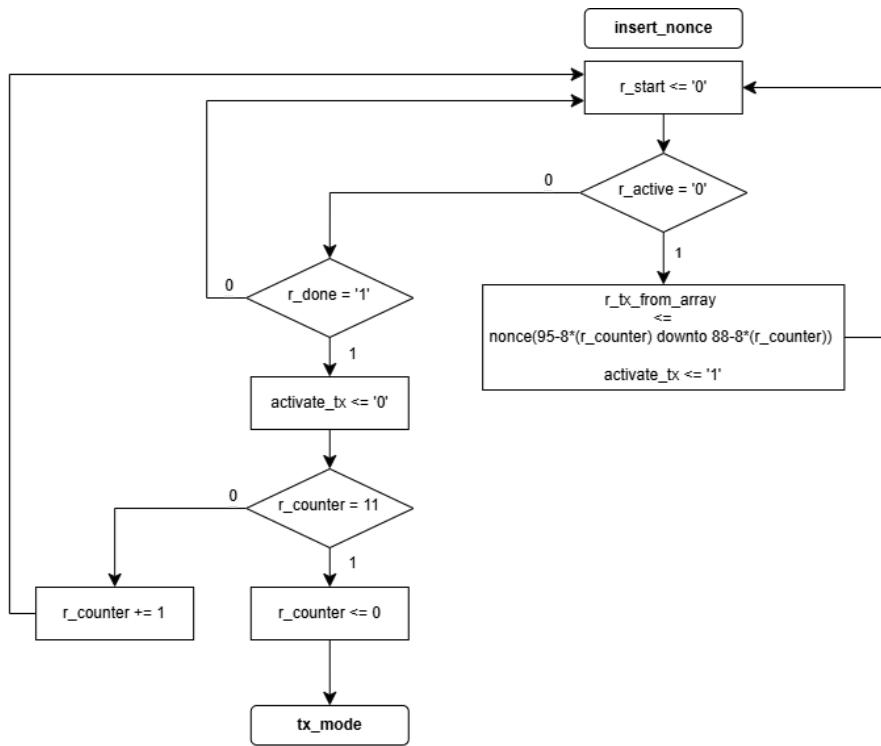
Gambar 3.1.5 ASM chart `encrypt_mode` state



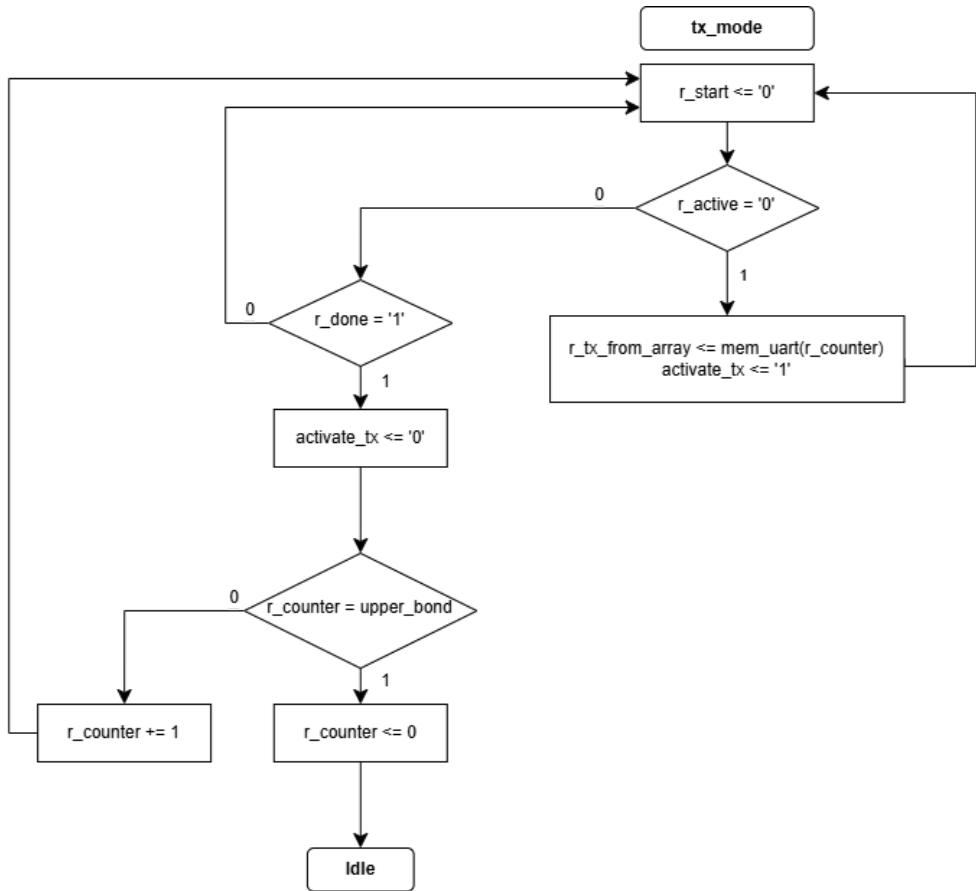
Gambar 3.1.6 ASM `key_nonce_maker_mode` Idle state



Gambar 3.1.7 ASM chart `insert_key` state

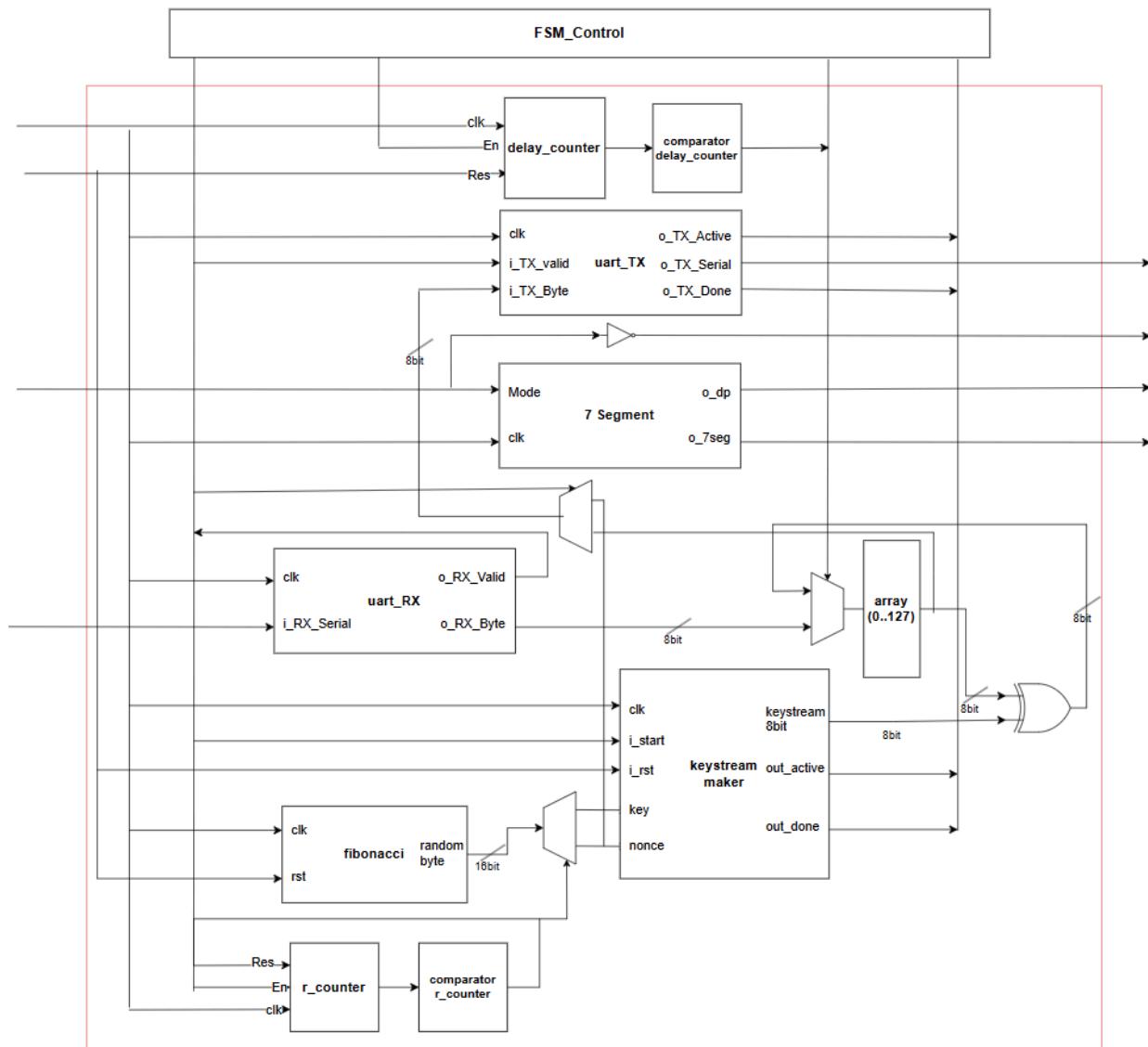


Gambar 3.1.8 ASM chart `insert_nonce` state



Gambar 3.1.9 ASM chart `tx_mode` state

Datapath



Gambar 3.2 Datapath rancangan akhir

Tabel 3.2 Keterangan komponen datapath

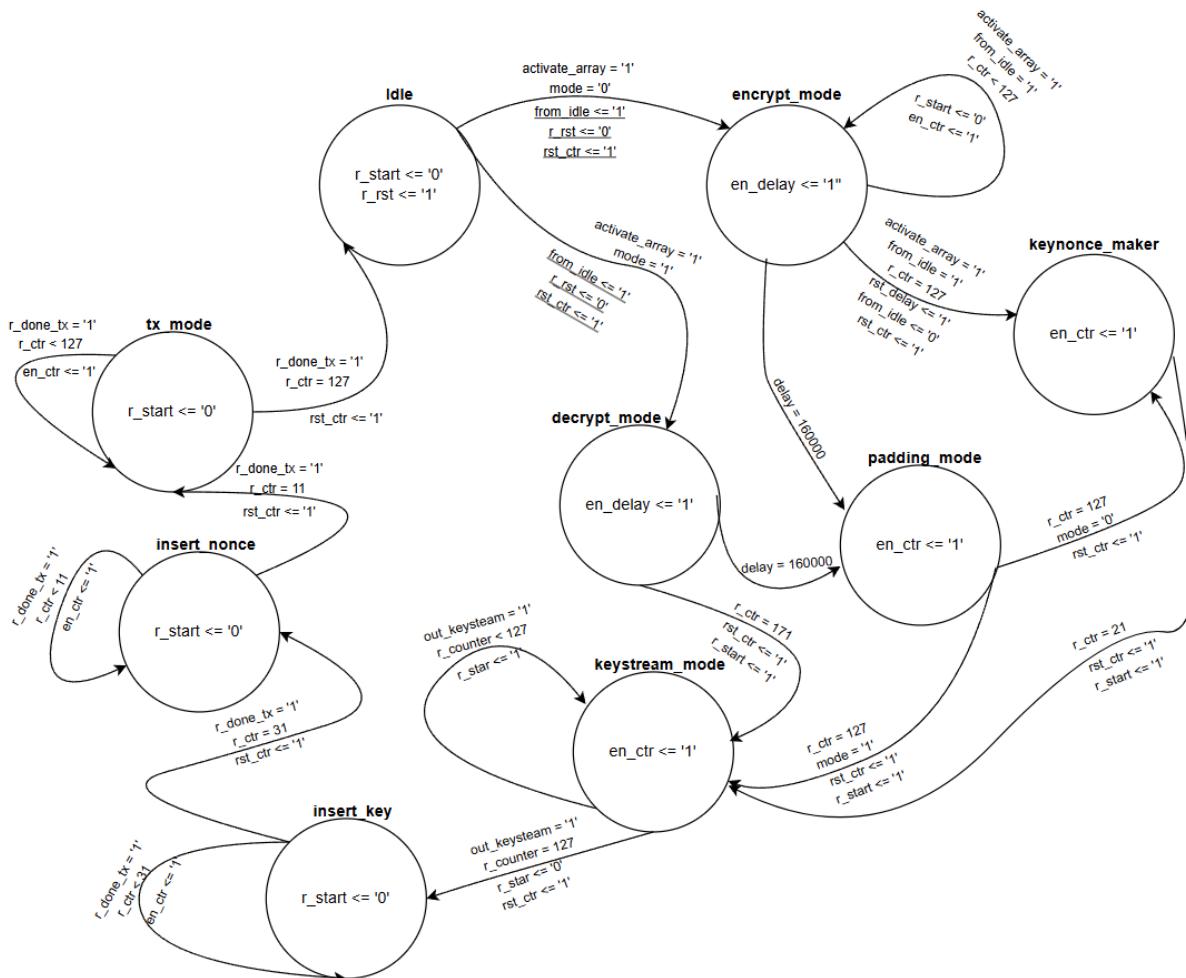
No.	Komponen	Input	Output	Keterangan
1.	UART RX	clk, <i>i_RX_Serial</i>	<i>o_RX_valid</i> , <i>o_RX_Byte</i>	Merupakan komponen uart receiver yang berfungsi untuk mengirimkan data menuju FPGA. Data yang akan dikirimkan adalah input text serta key dan nonce jika memilih mode dekripsi.

2.	UART TX	clk, i_TX_valid, i_TX_Byte	o_TX_Active, o_TX_Serial, o_TX_Done	Komponen uart transmitter untuk mengirimkan data dari FPGA. Data yang akan dikirimkan adalah key, nonce, dan text yang sudah di xor dengan key stream.
3.	Keystream Maker	i_start, i_RST, key, nonce, keystream8bit	out_active, out_done	Komponen yang mengandung algoritma chacha20. Pada komponen ini, input key dan nonce akan diproses bersama dengan konstanta dan counter untuk menghasilkan key stream 64 byte setiap blok nya. Namun pada komponen ini, key stream akan dikeluarkan 8 bit per clock.
4.	Fibonacci LFSR	clk, rst	random_byte	Fibonacci LFSR (Linear Feedback Shift Register) berfungsi untuk menghasilkan bilangan random 16 bite setiap clock nya. Bilangan 16 bit random ini akan digunakan sebagai key dan nonce untuk membuat key stream.
5.	7 Segment	mode, clk	o_dp, o_7segment	Menghasilkan nyala 7 segment pada FPGA untuk memberikan penanda mode yang sedang digunakan apakah enkripsi atau dekripsi.
6.	r_counter	clk, reset, enable	count	Menghitung berapa banyak iterasi yang sudah berjalan selama proses. Ini berfungsi sebagai kontrol pada sistem.
7.	Comparator r_counter	count	comp_flag	Memeriksa perhitungan yang sudah dilakukan oleh counter, dan output pertanda

				apakah sesuai jumlah perhitungan sudah sesuai dengan yang diinginkan.
8.	delay_counter	clk, reset, enable	delay_count	Menghitung waktu yang telah digunakan saat enkripsi ataupun dekripsi.
9.	Comparator delay_counter	delay_count	comp_flag	Menyamakan nilai delay_count dengan upper_delay. Jika sama, FSM akan berubah menjadi padding_mode.

Control FSM

Control FSM keseluruhan sistem



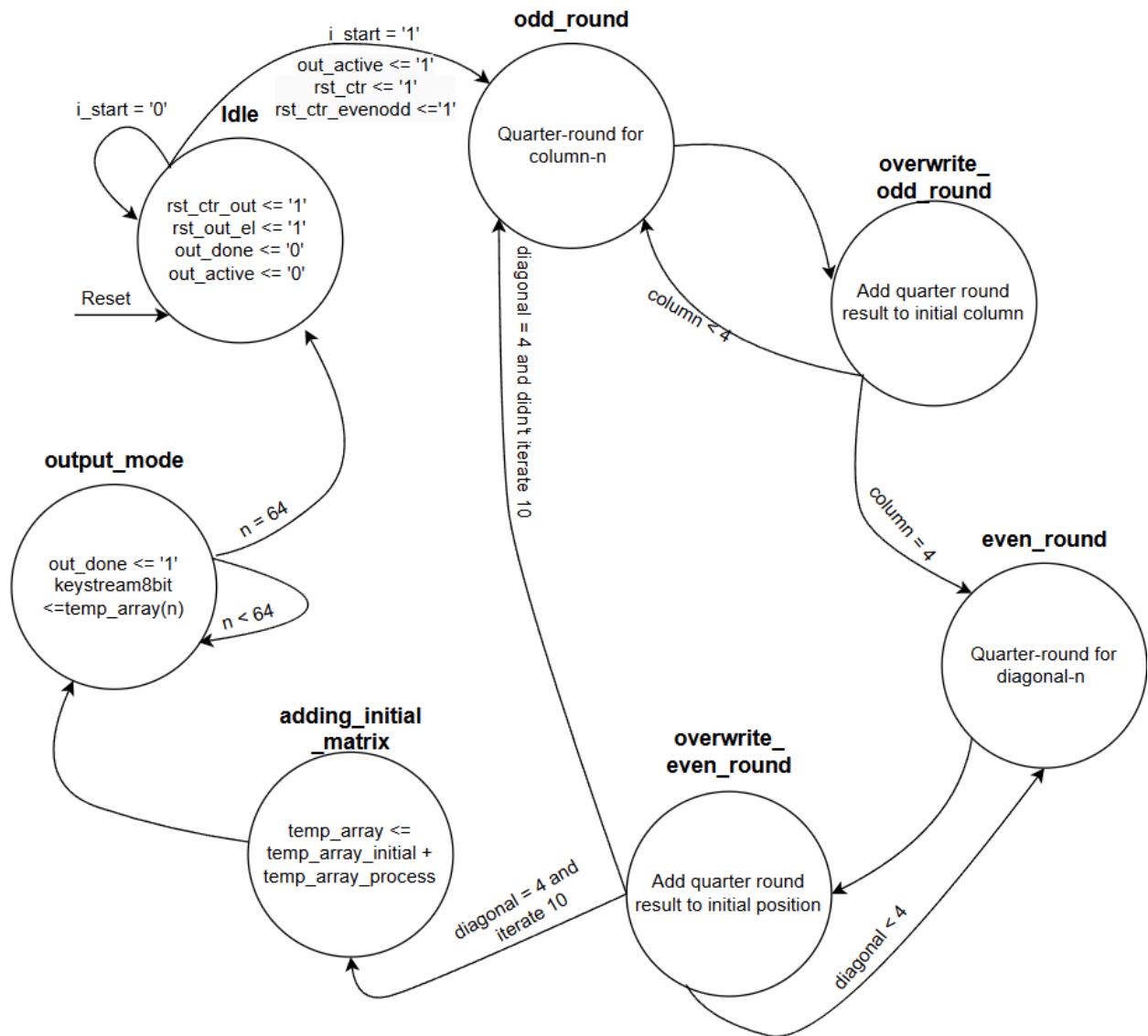
Gambar 3.3.1 Control FSM keseluruhan sistem

Kontrol untuk sistem keseluruhan dibagi menjadi 9 state. Kesembilan state dan apa yang dilakukannya di dalamnya adalah :

1. Idle : Pada state Idle, seluruh sistem masih dalam keadaan tidak aktif. Reset akan bernilai 1 dan start bernilai 0. Dari state ini akan beralih ke dua kemungkinan state berbeda bergantung dengan apa yang diinginkan oleh pengguna. Apakah mode enkripsi atau dekripsi, yang ditentukan oleh input mode mereka.
2. encrypt_mode : State ini merupakan tahapan dimana akan mengambil data berupa pesan yang akan dienkripsi. Pesan akan dimasukkan dari *whatsapp* dan akan diambil pesan terakhir yang terkirim. Pada state ini, mulai dilakukan perhitungan delay. Jika perhitungan delay sudah mencapai 160.000, maka akan berlanjut ke padding mode. decrypt_mode : Pada state ini, akan terjadi jika dari state Idle diberikan input mode dekripsi. Yang terjadi adalah pengambilan pesan yang terenkripsi, key dan nonce yang akan digunakan untuk membuka pesan tersebut. Jika counter sudah menunjukkan nilai 171, maka akan berlanjut pada mode pembuatan key stream dan proses XOR. Jika counter belum menyentuh nilai tersebut dan delay sudah terhitung 160.000, maka akan dimasukkan pada state padding mode.
3. decrypt_mode : Jika mode yang dipilih adalah mode dekripsi, yang dimasukkan adalah plain text, key serta nonce.
4. padding_mode : Merupakan Pada saat ini pesan yang dimasukkan kurang dari 128 karakter dan akan dilanjutkan dengan menambahkan “00000000” pada byte yang kosong. State berikutnya akan bergantung pada masukan mode yang diinginkan apakah enkripsi atau dekripsi. Jika diinginkan mode enkripsi, setelah ini akan berpindah pada state keynonce_maker. Sebaliknya, jika ini adalah mode dekripsi akan berlanjut pada state keystream_mode dan mengaktifkan pembuatan key stream. Kedua transisi state ini akan mereset counter agar kembali dari nol.
5. keynonce_maker : Merupakan suatu state untuk memasukkan bit random yang dihasilkan oleh Fibonacci LFSR ke dalam key dan nonce untuk pembentukan key stream. Proses ini hanya dilalui ketika mode yang diinginkan adalah enkripsi. Ketika counter sudah bernilai 22 (seluruh key dan noncesudah terisi) maka akan berpindah pada state keystream_mode sambil mereset nilai counter.
6. keystream_mode : State ini dapat berasal dari 3 state bergantung pada input yang dimasukkan apakah enkripsi atau dekripsi, serta apakah terdapat proses padding sebelumnya. Pada keystream ini, counter terus menghitung hingga 128. Dalam sekali pembuatan blok keystream, akan dihasilkan 64 byte dan setiap 8 bit nya akan di XOR dengan 8 bit input text yang ada. Setiap proses XOR akan terhitung sebagai pertambahan counter dan jika counter masih kurang dari 127, maka ia tetap berada

- di state tersebut serta membuat keystream kembali dengan mengaktifkan `r_start`. Jika sudah 127, akan masuk pada state `insert_key` sambil membuat `start` bernilai '0' dan mereset counter.
- 7. `insert_key` : Pada state ini akan dilakukan pengiriman key yang berjumlah 32 byte melalui TX. Jika sudah 32 byte terkirim (ditandai dengan counter), maka akan berlanjut pada state `insert_nonce` sambil mereset counter.
 - 8. `insert_nonce` : Mengirim nonce yang digunakan, berjumlah 12 byte yang dikontrol dengan counter.
 - 9. `tx_mode` : Untuk mengirimkan hasil XOR antara pesan masukan dengan key stream. Jika yang dimasukkan adalah pesan terenkripsi, maka pada proses ini akan dikirimkan pesan asli sebelum terkunci. Jika sudah 128 data dikirimkan, akan berpindah pada state `Idle`.

Control FSM penghasil Key Stream



Gambar 3.3.2 Control FSM `keystream_maker`

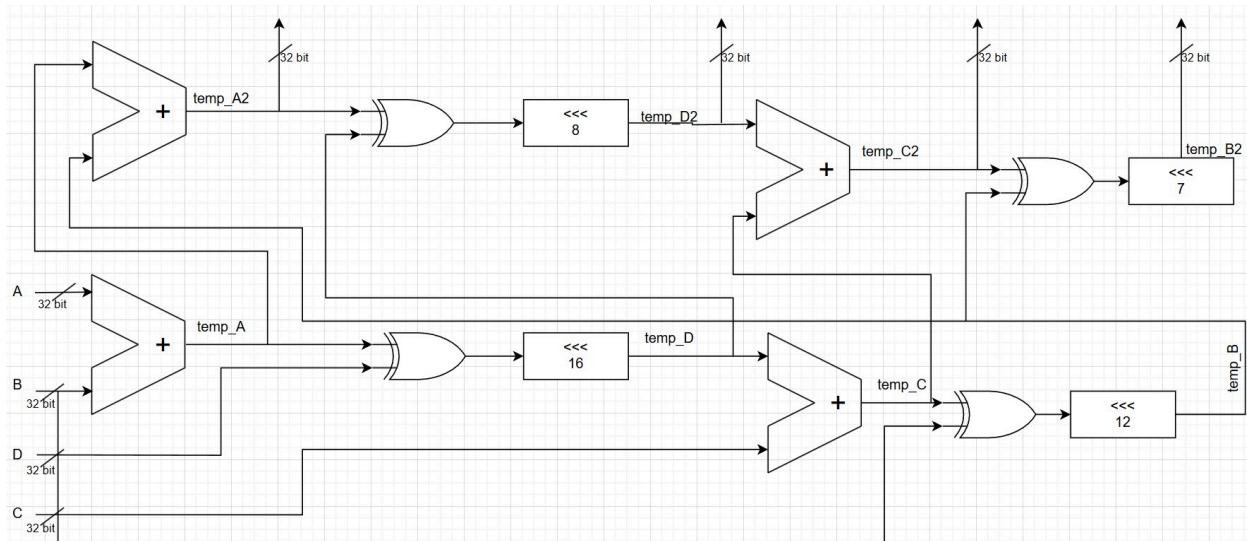
Key stream dibentuk dengan mengoperasikan key, nonce, counter dan konstanta menggunakan algoritma chacha20. Mulanya, seluruh input tersebut akan ditempatkan ke dalam matriks berukuran 4×4 masing-masing mengandung 4 byte data yang pada VHDL direpresentasikan dengan array.

A Konstanta	B Konstanta	C Konstanta	D Konstanta
E Key	F Key	G Key	H Key
I	J	K	L

Key	Key	Key	Key
M Counter	NNonce	ONonce	PNonce

Gambar 3.3.3 Matrix keystream

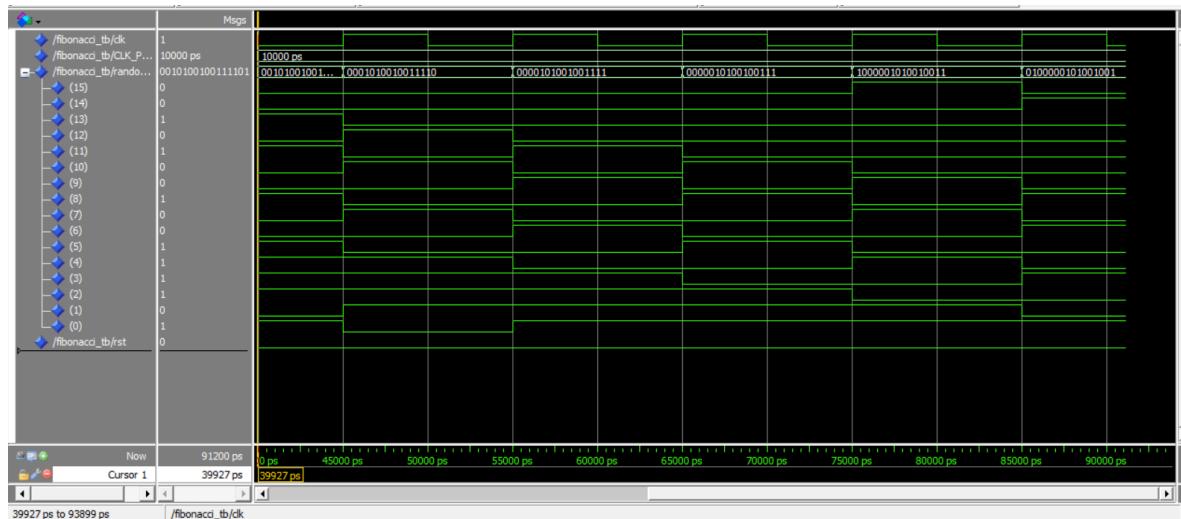
Matriks ini akan diolah dengan menggunakan algoritma chacha20, yang akan dibagi menjadi dua round, yaitu odd round dan even round. Sebanyak 4 elemen matriks akan diproses oleh quarter round seperti ditunjukkan pada gambar di bawah. Pada odd round akan diolah matriks dari kolom pertama hingga ke-empat. Kemudian pada even round akan diolah diagonal matriks dari kuning (A, F, K, P), biru (B, G, L, M), putih (C, H, I, N) dan hijau (D, E, J, O). Proses ini akan diulang hingga 10 kali untuk masing-masing round. Setelah itu akan ditambahkan dengan matriks awal sehingga dihasilkan keystream 64 byte.



Gambar 3.3.4 Diagram datapath quarter_round

Implementasi dan Simulasi

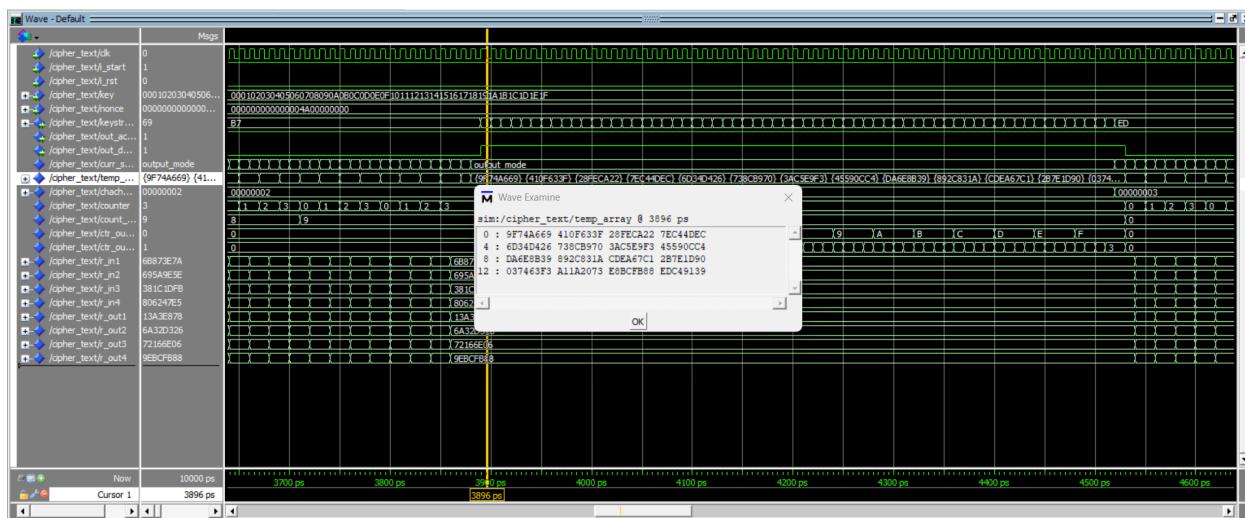
1. Fibonacci LFSR



Gambar 4.1 Waveform Simulasi Fibonacci LFSR

Dari waveform simulasi yang dilakukan, didapatkan output 16-bit random_byte berubah setiap satu *clock cycle* dengan nilai awalnya (1010010011110101) semakin bergeser ke kanan. Jika diberi sinyal *rst* = '1', random_byte akan kembali ke nilai awal. Nilai yang dihasilkan ini akan lebih random lagi jika menggunakan TRNG. Karena pada fibonacci, terdapat algoritma dan nilai awal.

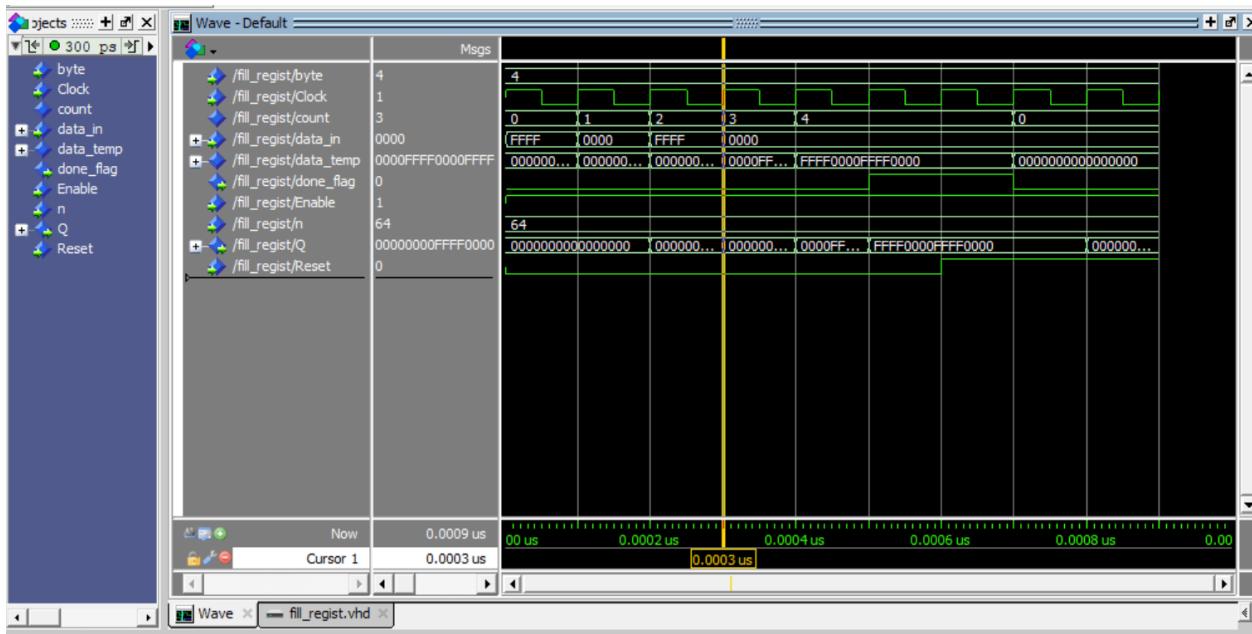
2. keystream_maker



Gambar 4.2 Waveform Simulasi keystream maker

Dengan input berupa key, nonce, start, reset dan clock maka pada bagian ini akan menghasilkan key stream yang dikeluarkan sebanyak 8 bit per clock nya. Pada gambar simulasi ditunjukkan bahwa ketika sudah melalui 10 kali putaran di even dan odd nya, dengan masing-masing 4 putaran sinyal bahwa sistem ini sudah selesai menghasilkan keystream 64 byte akan aktif. Kemudian dikeluarkan sebanyak 8 bit per clock nya.

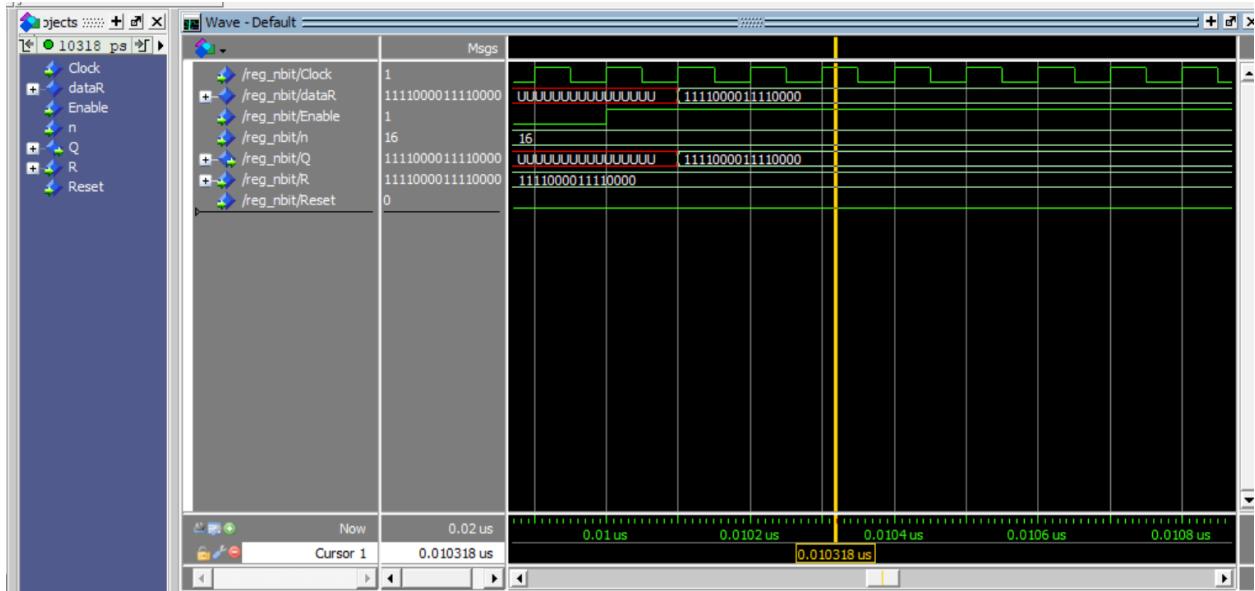
3. shiftreg_nbit



Gambar 4.3 Waveform Simulasi shiftregister_nbit

Bagian ini berfungsi untuk menyimpan data yang dimasukkan ke dalam register hingga register terisi penuh. Ketika penuh, akan mengeluarkan sinyal done_flag. Jika di reset, akan memberikan nilai '0' untuk register tersebut. Simulasi digunakan dengan menggunakan shift register yang menyimpan 64 bit dan diberi input 18 bit.

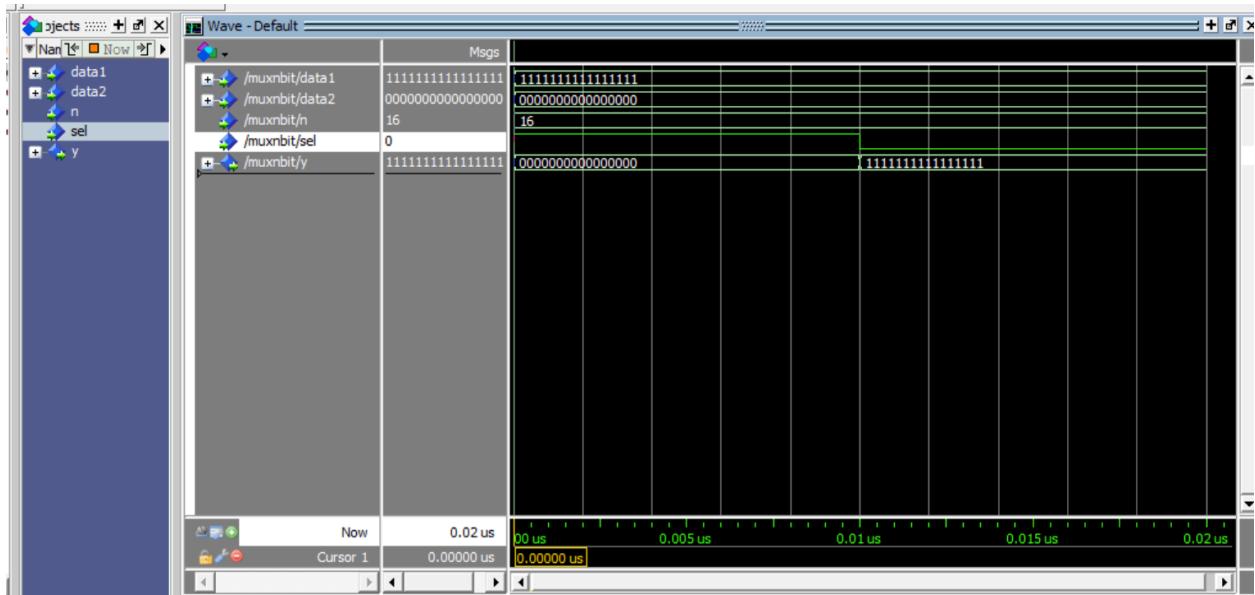
4. register n bit



Gambar 4.4 Waveform Simulasi register_nbit

Register akan menerima input jika enable bernilai ‘1’. Pada simulasi tersebut, ketika nilai enable masih 0, meskipun sudah diberikan input akan, output masih tidak berubah. Namun ketika enable dijadikan ‘1’, output akan sama dengan input yang masuk.

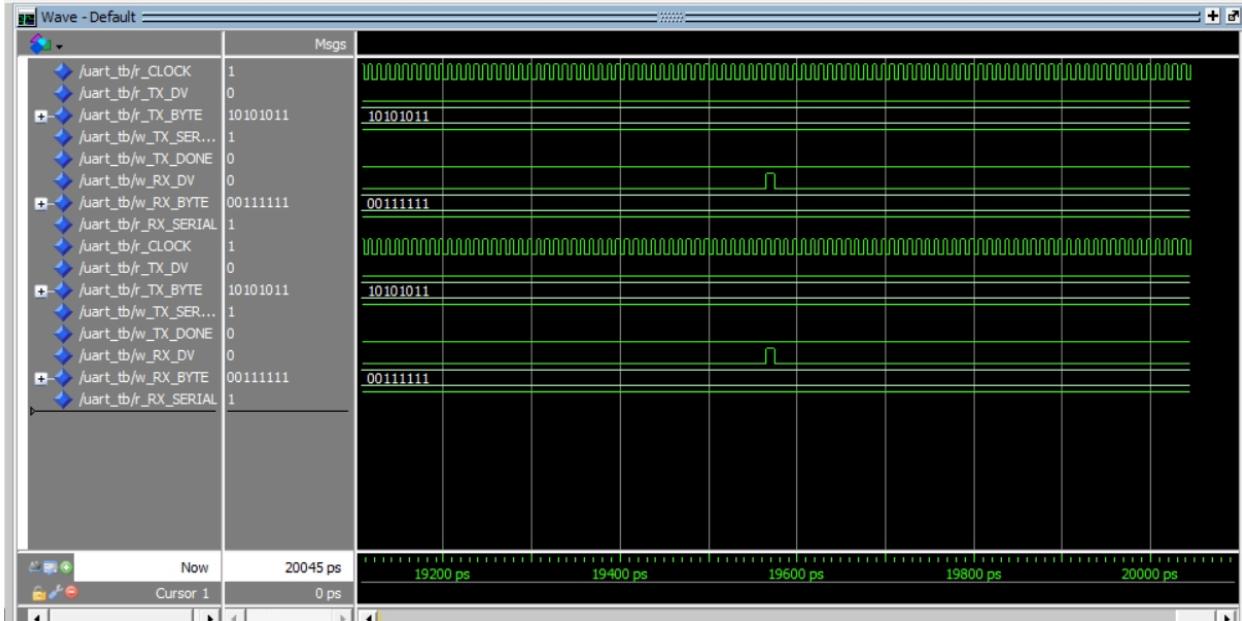
5. multiplexer 2 to 1



Gambar 4.5 Waveform Simulasi mux2_to_1

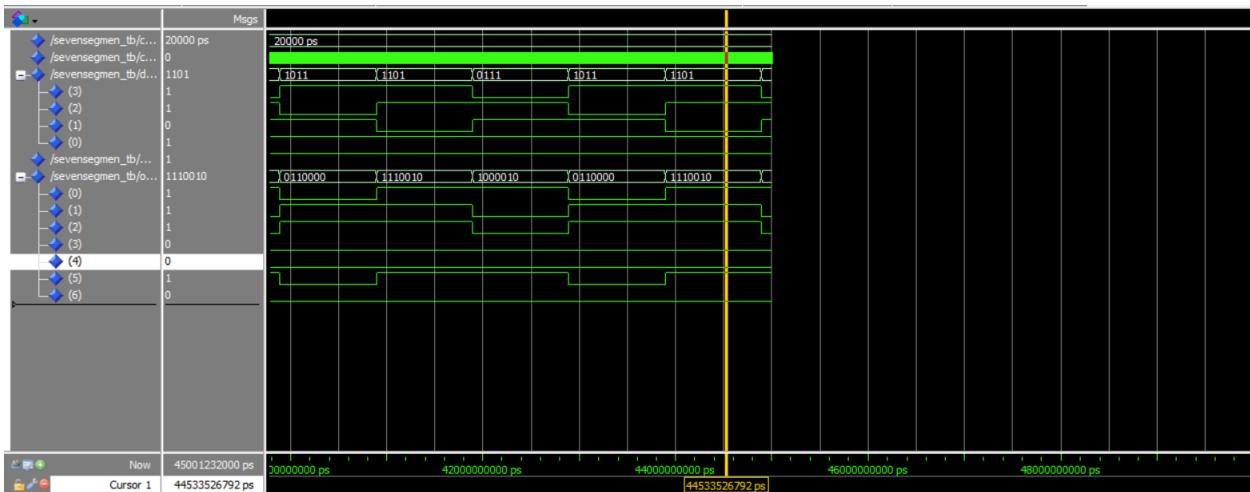
Ketika selektor bernilai ‘1’, output yang dihasilkan adalah data2. Sedangkan ketika selektor bernilai ‘0’, output yang dihasilkan adalah sama dengan data1.

6. Uart



Gambar 4.6 Waveform Simulasi UART

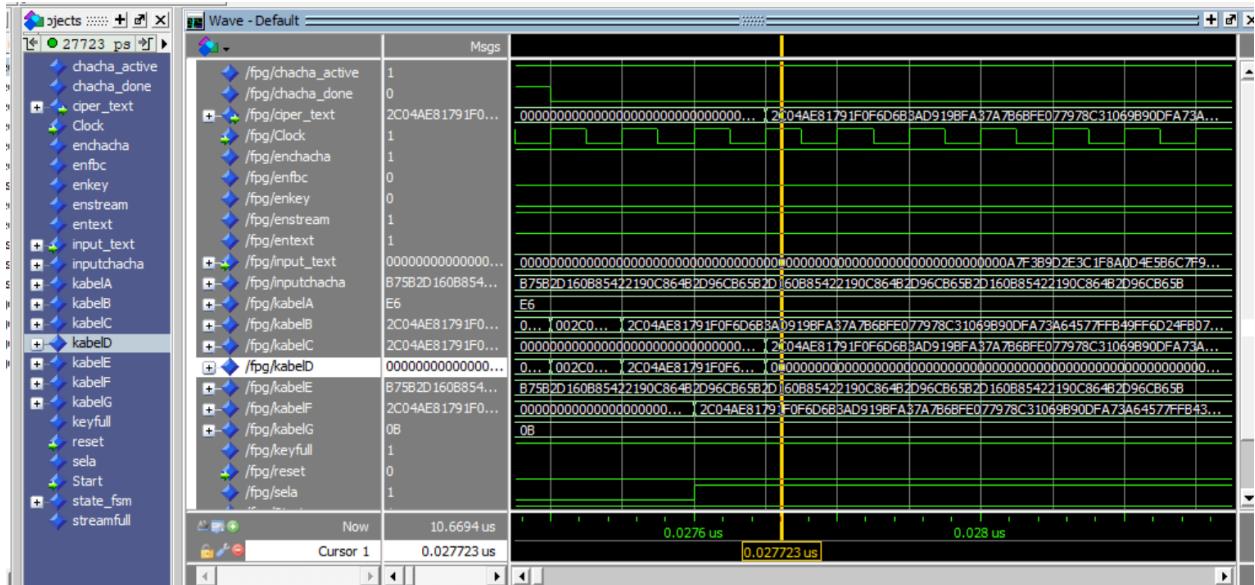
7. Sevsegmen



Gambar 4.7 Waveform Simulasi sevsegmen

Value output o dan dp sevsegmen bergantung pada current state (dp1, dp2 atau dp3) dan input mode. Sinyal ctr dengan range 0 to 50001 memastikan terjadinya perubahan state setiap 50001 clock cycles.

8. Desain Awal tanpa UART



Gambar 4.8 Waveform implementasi rancangan awal tanpa UART

Tabel 4.1 Implementasi dan Simulasi VHDL tiap Entity

No.	Entity	VHDL	Simulasi
1.	mux	Ya	Ya
2.	register_nbit	Ya	Ya
3.	shiftreg_nbit	Ya	Ya
4.	quarter_round	Ya	Ya
5.	sevensegment (higher-level)	Ya	Ya
6.	Fibonacci LSFR (higher-level)	Ya	Ya
7.	uart_rx (higher_level)	Ya	Ya
8.	uart_tx (higher_level)	Ya	Ya
9.	keystream_maker (higher_level)	Ya	Ya
10.	uart (top-level)	Ya	Ya

Implementasi FPGA dan Pengujian

- Enkripsi - Dekripsi 1

Key: A4 F5
Nonce: A4 F5 A4 F5 A4 F5 A4 F5 A4 F5 A4 F5
Chacha Hex: 4B 65 6C 6F 6D 70 6F 6B 20 31 34 20 73 69 73 74 65 6D 20 64 69 67 69 74 61 6C 20 32 30 32 34 20 2D 2
0 32 30 32 34 20 49 6E 73 74 69 74 75 74 20 54 65 6B 6E 6F 6C 6F 67 69 20 42 61 6E 64 75 6E 67 2C 20 49 6E 64 6F
6E 65 73 69 61 00
00
Text: Kelompok 14 sistem digital 2024 - 2024 Institut Teknologi Bandung, Indonesia

- Enkripsi - Dekripsi 2

Pada uji yang sudah dilakukan melalui python terlihat bahwa untuk proses enkripsi-dekripsi akan menghasilkan pesan yang benar dengan memasukkan key dan nonce yang benar. Proses enkripsi hanya memberikan input text yang akan di enkripsi dan diberikan output berupa hexadesimal text hasil enkripsi, key dan nonce. Jika ingin membuka pesan terkunci tersebut, cukup memasukkan pesan terkunci, key dan nonce yang digunakan untuk mengenkripsi.

Lampiran

Kontribusi

NIM>Nama	% Kontribusi I	% Kontribusi II	% Kontribusi III	% Kontribusi IV
13223112 - Ahmad Fatur Rohman	10 % Bentuk Kontribusi: <ul style="list-style-type: none">• Komponen Fibonacci• UART TX• Perancangan awal• Integrasi rancangan awal tanpa UART	% -	30% Bentuk Kontribusi: <ul style="list-style-type: none">• FSM rancangan akhir• FSM key stream maker• Datapath rancangan awal-akhir• Laporan	20 % Bentuk Kontribusi: Hadir kerja kelompok
13223039 - Dimas Anggiat	10 % Bentuk Kontribusi: <ul style="list-style-type: none">• Simulasi cipher text• Mux nbit 2to1	% -	15% Bentuk Kontribusi: <ul style="list-style-type: none">• Power point• Datapath rancangan awal• Diagram quarter round	20 % Bentuk Kontribusi: Hadir kerja kelompok
13223110 - Julian Benedict	10 % Bentuk Kontribusi: <ul style="list-style-type: none">• Simulasi VHDL : UART• Shift register n-bit	40 % Bentuk Kontribusi: <ul style="list-style-type: none">• Kode Python untuk pengambilan pesan dari whatsapp	15% Bentuk Kontribusi: <ul style="list-style-type: none">• Video Dokumentasi• Power Point• FSM rancangan awal	20 % Bentuk Kontribusi: Hadir kerja kelompok
13223038 - Kyla Asha Mulani	10 % Bentuk Kontribusi: <ul style="list-style-type: none">• Simulasi VHDL : 7segment & fibonacci	- Bentuk Kontribusi: <ul style="list-style-type: none">• Datapath rancangan akhir	30 % Bentuk Kontribusi: <ul style="list-style-type: none">• Datapath rancangan akhir	20 % Bentuk Kontribusi: Hadir kerja kelompok

			<ul style="list-style-type: none"> • ASM rancangan akhir • Laporan 	
13223040	60%	60%	10%	20 %
- Mohamma d Najmutsaq ib	<p>Bentuk Kontribusi:</p> <ul style="list-style-type: none"> • Perancangan dan pembuatan key stream • Perancangan dan pembuatan UART • Pengendalian Driver CH340 	<p>Bentuk Kontribusi:</p> <ul style="list-style-type: none"> • Kode pendukung pengambilan pesan dengan python • Kode python implementasi UART • Pengujian port UART 	<p>Bentuk Kontribusi:</p> <ul style="list-style-type: none"> • Video demonstrasi • Pembuatan diagram quarter round per clock 	<p>Bentuk Kontribusi: Hadir kerja kelompok</p>
Total	100%	100%	100%	100%

Link code VHDL

https://github.com/bukananda/sistem_digital

Dokumentasi

