Jurnal Coding, Sistem Komputer UNTAN ISSN : 2338-493X
Volume 04, No.1 (2016), hal 21-32

IMPLEMENTASI ALGORITMA KRIPTOGRAFI VERNAM *CIPHER* BERBASIS FPGA

[1]Mohammad Jumeidi, [2]Dedi Triyanto, [3]Yulrio Brianorman
[1][2][3]Jurusan Sistem Komputer, Fakultas MIPA Universitas Tanjungpura
Jl. Prof. Dr. H. Hadari Nawawi, Pontianak
Telp./Fax.: (0561) 577963
e-mail:
[1]mohammad_jumeidi@yahoo.co.id, [2]dedi.triyanto@siskom.untan.ac.id,
[3]yulrio.brianorman@siskom.untan.ac.id

Abstrak

Keamanan data menjadi salah satu isu penting dalam perkembangan teknologi informasi saat ini. Salah satu cara yang dapat dilakuan untuk menjaga keamanan data adalah dengan menggunakan ilmu kriptografi. Untuk tujuan yang khusus, implementasi kriptografi pada perangkat keras memberikan kelebihan dalam hal kecepatan dan penghematan sumber daya. Salah satu perangkat keras yang mendukung hal tersebut adalah FPGA. Pada penelitian ini dilakukan implementasi algoritma kriptografi berbasis FPGA menggunakan Altera UP2 Education Kit. Rancangan sistem dibuat dengan menggunakan bahasa VHDL. Sistem ini dapat melakukan proses enkripsi-dekripsi. Proses enkripsi-dekripsi dilakukan dengan memasukan data pada sistem menggunakan switch pada Altera UP2 Education Kit. Input dan output sistem ditampilkan dengan menggunakan seven segment. Hasil dari penelitian ini adalah sistem dapat melakukan proses enkripsi-dekripsi data dengan ukuran data 8 bit.

Kata Kunci: *Keamanan Data, Kriptografi, Vernam Cipher, FPGA, VHDL*

1. PENDAHULUAN

Teknologi saat ini terus berkembang dan mengalami kemajuan yang sangat pesat. Kemajuan teknologi ini mengakibatkan arus pertukaran data dari satu tempat ke tempat lain menjadi lebih mudah. Berbagai macam data dapat disampaikan melalui teknologi saat ini, baik data yang sifatnya terbuka maupun rahasia. Data yang bersifat rahasia perlu dijaga kerahasiaannya agar data tersebut aman dan tidak disalah gunakan. Untuk menjaga keamanan data diperlukan suatu metode tertentu. Salah satu metode yang dapat digunakan untuk menangani masalah tersebut adalah kriptografi. Implementasi algoritma kriptografi dapat dilakukan dalam bentuk perangkat lunak maupun perangkat keras [6]. Implementasi pada perangkat keras dapat dilakukan dengan berbasis mikrokontroler maupun *programmable logic device* seperti FPGA. Menurut randal dan Panos (2002), teknologi FPGA memberikan gabungan keuntungan yaitu performansi dari ASIC dan fleksibilitas yang dimiliki *processor*. Ada dua tujuan utama dalam penggunaan FPGA sebagai *platform* komputasi yaitu penghematan sumber daya dan pemaksimalan kecepatan pemrosesan. Berdasarkan hal tersebut maka teknologi FPGA dapat menjadi alternatif *platform* untuk mengimplementasikan komputasi khusus dalam hal ini adalah implementasi algoritma kriptografi. Dari implementasi ini diharapkan dapat memberikan alternatif bentuk implementasi algoritma kriptografi yang hemat sumber daya dan cepat dalam pemrosesannya.

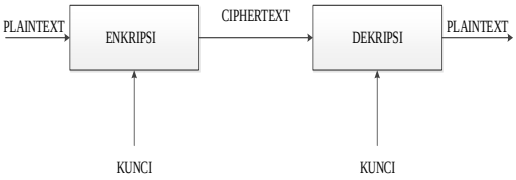
2. TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi adalah ilmu pengetahuan sekaligus seni menjaga pesan [5]. Berdasarkan termanologinya kriptografi berasal dari bahasa Yunani yaitu *kryptos* yang berarti menyembunyikan dan *graphein* yang berarti menulis, sehingga kriptografi dapat didefinisikan sebagai ilmu yang mengubah informasi dari keadaan/bentuk normal (dapat dipahami) menjadi bentuk yang tidak dapat dipahami.

 ISSN : 2338-493X

Secara sederhana proses kriptografi adalah seperti Gambar 1.



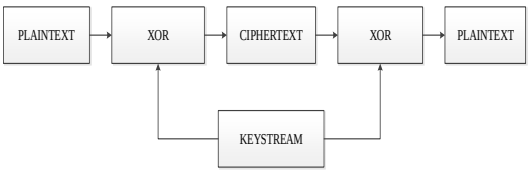
Gambar 1. Proses Enkripsi dan Dekripsi

2.2 Algoritma Vernam *Cipher*

Vernam *cipher* merupakan algoritma kriptografi yang ditemukan oleh Mayor J. Maugborne dan G. Vernam. Algoritma Vernam *cipher* diadopsi dari *one time pad cipher*, dimana dalam hal ini karakter diganti dengan bit (0 atau 1). Dengan kata lain, vernam *cipher* merupakan versi lain dari *one-time pad cipher* [4].

Algoritma kriptografi vernam *cipher* merupakan algoritma kriptografi berjenis *symmetric key*. Kunci yang digunakan untuk melakukan enkripsi dan dekripsi menggunakan kunci yang sama. Dalam melakukan proses enkripsi, algoritma vernam *cipher* menggunakan cara *stream cipher* dimana *cipher* berasal dari hasil operasi XOR antara bit *plainteks* dan bit key [6].

Pada *cipher* aliran, bit hanya mempunyai dua buah nilai, sehingga proses enkripsi hanya menyebabkan dua keadaan pada bit tersebut, yaitu berubah atau tidak berubah. Dua keadaan tersebut ditentukan oleh kunci enkripsi yang disebut dengan aliran-bit-kunci (*keystream*). Secara sederhana proses enkripsi dan dekripsi algoritma vernam *cipher* dapat adalah pada Gambar 2.

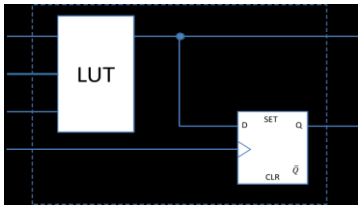


Gambar 2. Proses Enkripsi dan Dekripsi Algoritma Kriptografi Vernam *Cipher*

2.3 FPGA

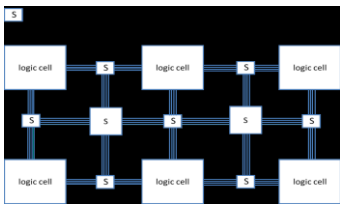
Field programmable gate array (FGPA) merupakan perangkat pengembangan berupa *chip* yang terdiri dari ribuan atau jutaan transistor yang terkoneksi dan dapat memiliki fungsi logika tertentu. FPGA adalah perangkat semikonduktor yang dapat diporgram/dikonfigurasi ulang setelah proses manufaktur. FPGA memungkinkan *costumer* atau *programmer* untuk dapat membuat fitur atau fungsi program, beradaptasi dengan standar baru, dan mengkonfigurasi ulang *hardware* untuk aplikasi tertentu bahkan setelah program telah terinstal di board. FPGA juga bisadigunakan untuk mengimplementasikan berbagai *logic function* [3].

FPGA mengandung komponen yang disebut *logic cell* dan *programmable switch*. *Logic cell* dapat dikonfigurasi atau deprogram untuk menjalankan kompleks, atau gerbang logika sederhana seperti AND dan OR. Setiap *logic cell* terdiri atas satu *look up table* dan satu D-*flip flop*. Setiap *logic cell* saling terkait antar satu sama lain dan dihubungkan dengan *programmable swtich* [3]. Adapun gambaran *logic cell* pada FGPA dapat lihat seperti pada Gambar 3.



Gambar 3. *logic Cell*

Konfigurasi FPGA ditentukan dengan menggunakan *hardware description language* (HDL). Terdapat dua HDL yang umum digunakan yaitu *Verilog dan Very High-Speed Integrated Circuit* (VHSIC) HDL. Pada pertengahan tahun 1980-an Departemen Pertahanan AS dan IEEE mensponsori perkembangan *hardware description language* dengan tujuan untuk mengembangkan *high-speed integrated circuit*. Sekarang bahasa VHDL ini telah menjadi salah satu bahasa standar industri yang digunakan untuk merancang dan menjelaskan sistem digital [3]. Struktur konseptual dari FPGA dapat dilihat pada Gambar 4.



Gambar 4. Struktur Konseptual Dari FPGA

 ISSN : 2338-493X

Terdapat empat langkah yang dilakukan dalam mengembangkan implementasi pada FPGA [3]. Keempat langkah tersebut yaitu:

1. Mendesain sistem dan menurunkannya ke *Hardware Description Language* (HDL) *file*.
2. Mengembangkan t*estbench* dengan HDL dan membuat simulasi pada *Register Transfer Lavel* (RTL).
3. Men-*synthesis* dan mengimplementasikan HDL menjadi gerbang-gerbang logika.
4. Terakhir adalah men-*generate* dan men-*download programming file.*

2.4 VHDL

Very high speed integrated circuit HDL (VHDL) merupakan bahasa yang digunakan untuk memodelkan perangkat keras digital. VHDL digunakan untuk pengembangan ASIC (*Application Specific Integrated Circuits*). VHDL dapat dipakai untuk memodelkan perilaku sistem terbebas dari teknologi targetnya. Termasuk di dalamnya untuk memodelkan mikrokontroler, koder, model perilaku mikroporsesor, dan perangkat RAM [1].

Seperti pada bahasa pemograman yang lainnya, VHDL juga memiliki struktur dan aturan yang harus dipatuhi. Setiap unit rancangan VHDL terdiri dari suatu deklarasi *entity* dan satu atau lebih *architecture*. Setiap *architecture* mendefinisikan implementasi atau model yang berbeda dari suatu unit rancangan yang diberikan. Definisi *entity* menetapkan semua masukan dan keluaran dari modul, serta setiap *generic* parameter yang digunakan oleh implementasi yang berbeda dari suatu modul [1]. Format deklarasi *entity* pada VHDL adalah sebagai berikut:

```
entity name is
    port( port definition list );-- input/output signal
ports
    generic( generic list); -- optional generic list
end name;
```

2.5 Gerbang Logika Dasar

Gerbang logika merupakan diagram blok symbol rangkaian digital yang memproses sinyal masukan menjadi sinyal keluaran dengan perilaku tertentu. Terdapat tiga tipe dasar gerbang logika yaitu AND, OR, dan NOT. Masing-masing gerbang dasar dapat dikombinasikan satu dengan yang lainnya membentuk gerbang turunan yaitu NAND (NOT AND), NOR (NOT OR), XOR (Exclusive OR), dan XNOR (Exclusive NOT OR). Tiap-tiap gerbang memiliki sifat yan berbeda atau logika proses yang berbeda. Perbedaan ini dapat ditunjukkan dengan kombinasi keluaran yang digambarkan dalam tabel kebenaran [1]. Tabel kebenaran menunjukan fungsi gerbang logika yang berisi kombinasi masukan dan keluaran. Dalam tabel kebenaran ditunjukkan hasil kebenaran setiap kombinasi yang mungkin dari sinyal masukan pada gerbang logika [1].

2.6 Sistem Bilangan

Sistem bilangan adalah notasi untuk merepresentasikan bilangan. Pada sistem digital sistem bilangan yang digunakan adalah bilangan biner dan heksadesimal yaitu basis 2 dan 16. Pada kehidupan sehari-hari sistem bilangan yang banyak digunakan adalah bilangan desimal yaitu bilangan basis 10. Sitem bilangan lain yang juga digunakan adalah bilangan oktal yaitu sistem bilangan basis 8 [1]. Sistem hexadecimal memiliki 16 kemungkinan nilai (symbol digit). Nilai hexadecimal adalah 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, dan F. Tabel 1 merupakan hubungan antara hexadesimal, desimal, dan biner.

Tabel 1. Konversi Heksadesimal, Oktal, dan Biner

Hexadesimal	Desimal	Biner
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011

 ISSN : 2338-493X

Hexadesimal	Desimal	Biner
C	12	1100
D	13	1101
E	14	1110
F	15	1111

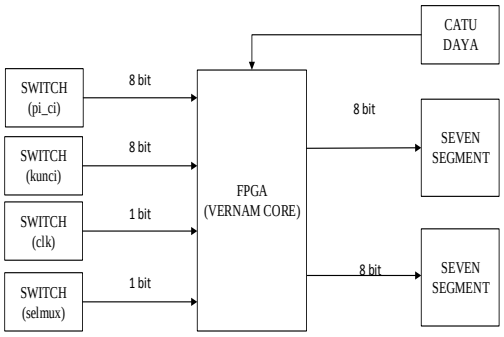
3. METODOLOGI PENELITIAN

Metodologi dalam penelitian ini meliputi keseluruhan tahapan yang dilakukan dalam melaksanakan penelitian. Adapun tahapan yang dilakukan yaitu identifikasi masalah, studi pustaka, perencanaan penelitian, analisis kebutuhan, integrasi, pengujian, serta analisa dan kesimpulan. Pada peneltian ini sistem yang dibuat mengacu kepada referensi yang sudah ada.

4. PERANCANGAN

4.1 Perancangan Prinsip Kerja Sistem

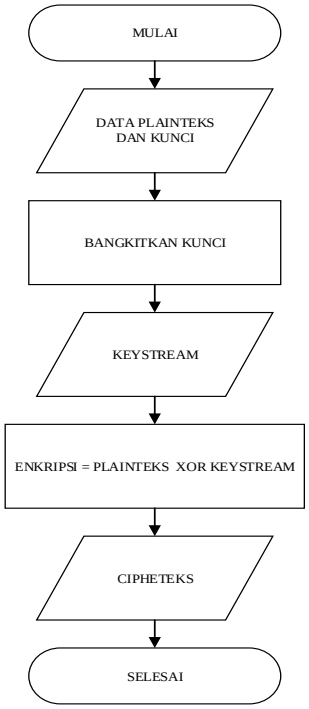
Pada penelitan ini dibuat sebuah rancangan sistem berupa implementasi algoritma kriptografi vernam *cipher* berbasis FPGA yang dapat melakukan proses enkripsi/dekripsi data dengan lebar data 8 bit. Untuk melakukan proses enkripsi/dekripsi, pada penelitian ini digunakan kunci dengan lebar 8 bit. Gambar 5 berikut ini merupakan diagram blok perancangan sistem dari implementasi algoritma kriptografi vernam *cipher* berbasis FPGA.



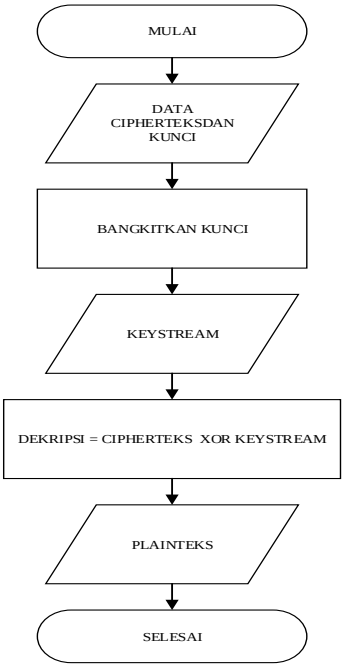
Gambar 5. Diagram Blok Sistem

4.1.1 Proses Enkripsi/Dekripsi Algoritma Vernam *Cipher*

Algoritma kriptografi vernam *cipher* merupakan algoritma simetri yang dalam proses enrkipsi dan dekripsinya menggunakan kunci yang sama. Proses enkripsi dan dekripsi algoritma kriptografi vernam *cipher* dapat lihat seperti pada Gambar 6 dan Gambar 7.



Gambar 6. Diagram Alir Proses Enkripsi Algoritma Vernam *Cipher*



Gambar 7. Diagram Alir Proses Dekripsi Algoritma Vernam *Cipher*

Dari diagram alir pada Gambar 6 dan Gambar 7 dapat dilihat bahwa proses enkripsi maupun dekripsi dilakukan dengan menggunakan operasi xor, dimana pada proses enkripsi operasi xor

 ISSN : 2338-493X

dilakukan pada *plainteks* dan aliran bit kunci (*keystream*) untuk mendapatkan *cipherteks*. Pada proses dekripsi operasi xor dilakukan pada *cipherteks* dan aliran bit kunci (*keystream)* untuk mendapatkan *plainteks*. *Keystream* merupakan aliran bit kunci yang dibangkitkan oleh pembangkit kunci (*key generator*). Pada Penelitian ini digunakan pembangkit kunci berupa *register* geser dengan umpan balik.

4.1.2 Tampilan Input-Output Pada *Seven segment*

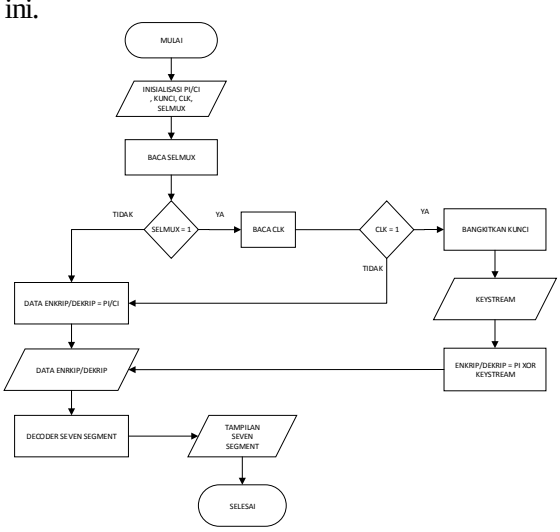
Seven segment yang digunakan dalam penelitian ini sebanyak dua buah. Pada penelitian ini *seven segment* digunakan untuk menampilkan bilangan heksadesimal dari *plainteks*/*cipherteks*. Untuk menampilakan 8 bit *plainteks*/*cipherteks* pada dua buah *seven segment* tersebut, dirancang suatu penerjemah atau *decoder seven segment*. Proses penerjemahan 8 bit *plainteks*/*cipherteks* ke dalam dua buah *seven segment* dilakukan dengan cara membagi 8 bit *plainteks*/*cipherteks* menjadi dua buah kelompok bit yang masing-masing terdiri dari 4 bit. Dua buah kelompok bit ini kemudian digunakan sebagai pengatur dari *output* pada masing-masing *seven segment*. Tabel 2 menunjukkan *decoder seven segment* dalam penelitan ini.

Tabel 2. Decoder *Seven segment*

No	Kelom-pok Bit	7 *Segment*								*Display Seven segment*
		a	b	c	d	e	f	g	h	
1	0000	0	0	0	0	0	0	1	1	0
2	0001	1	0	0	1	1	1	1	1	1
3	0010	0	0	1	0	0	1	0	1	2
4	0011	0	0	0	0	1	1	0	1	3
5	0100	1	0	0	1	1	0	0	1	4
6	0101	0	1	0	0	1	0	0	1	5
7	0110	0	1	0	0	0	0	0	1	6
8	0111	0	0	0	1	1	1	1	1	7
9	1000	0	0	0	0	0	0	0	1	8
10	1001	0	0	0	0	1	0	0	1	9
11	1010	0	0	0	1	0	0	0	1	A
12	1011	1	1	0	0	0	0	0	1	b
13	1100	0	1	1	0	0	0	1	1	C
14	1101	1	0	0	0	0	1	0	1	d
15	1110	0	1	1	0	0	0	0	1	E
16	1111	0	1	1	1	0	0	0	1	F

4.1.3 Alur Kerja Sistem

Alur kerja sistem pada penelitian ini menggambarkan cara kerja sistem secara keseluruhan mulai dari proses *input* data sampai tampilan akhir pada *seven segment*. Alur kerja sistem dimulai dengan memberikan *input* pada sistem yaitu *plainteks*/*cipherteks* (pi_ci), *input* kunci (key), *input* tampilan *seven segment* (selmux), dan *input* proses enkripsi/dekripsi (clk). Sistem membaca *input* tampilan *seven segment* (selmux) dan membandingkannya dengan kondisi yang telah ditentukan untuk melakukan proses selanjutnya dalam menampilkan *output* pada *seven segment*. Secara lengkap alur kerja sistem pada implementasi algoritma kriptografi vernam *cipher* berbasis FPGA dalam penelitian ini dapat dilihat seperti pada Gambar 8 berikut ini.



Gambar 8. Diagram Alir Kerja Sistem

4.2 Perancangan Arsitektur Hardware Algoritma Vernam *Cipher*

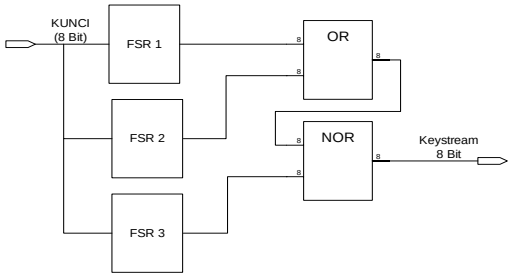
4.2.1 Perancangan Pembangkit Kunci

Tahap pertama dalam perancangan arsitektur *hardware* algoritma kriptografi vernam *cipher* pada penelitan ini adalah perancangan arsitektur pembangkit kunci. Pada tahap ini dibuat arsitektur pembangkit kunci (*keystream generator*) dengan menggunakan *register* geser yang dikombinasikan dengan fungsi umpan balik (*feedback shift register*). *Register* geser dan fungsi umpan balik bertugas untuk mengacak kunci, sehingga didapat kunci lain yang kemudian digunakan sebagai *keystream* dalam proses enkripsi/dekripsi.

Terdapat tiga buah *feedback shift register* (FSR) pada perancangan arsitektur *hardware*

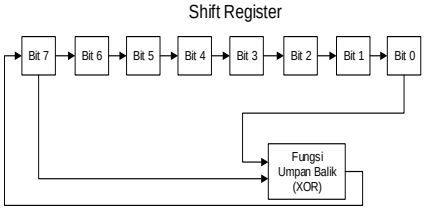
 ISSN : 2338-493X

pembangkit kunci dalam penelitian ini. Ketiga buah FSR tersebut memilki fungsi umpan balik yang sama yaitu menggunakan fungsi umpan balik berupa operasi XOR. Keluaran (*output*) dari hasil operasi pada tiga buah FSR ini menghasilkan sebuah kunci baru yang disebut *keystream*. Rancangan aristektur pembangkit kunci pada penelitan ini dapat dilihat seperti Gambar 9. Pada penelitian ini FSR mendapatkan *input* dari sebuah kunci yang sama. *Output* dari FSR1 dioperasikan dengan *output* FSR2 dengan menggunakan operator OR. Hasil dari operasi tersebut kemudian dioperasikan kembali dengan *output* dari FSR3 dengan menggunakan operator NOR. Keluaran dari operasi ini menghasilkan sebuah kunci baru (*keystream*).

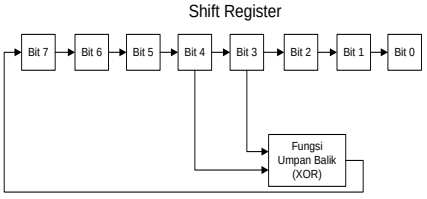


Gambar 9. Arsitektur Pembangkit Kunci

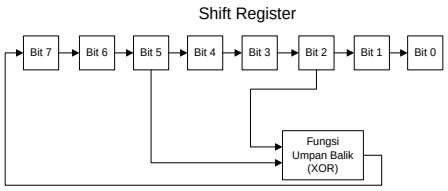
Untuk mendapatkan *output* dari FSR1, FSR2, dan FSR3, selanjutnya dirancang alur kerja dari FSR. Adapun alur kerja dari FSR pada penelitian ini ditunjukkan pada Gambar 10, 11, dan 12.



Gambar 10. Alur Kerja FSR 1



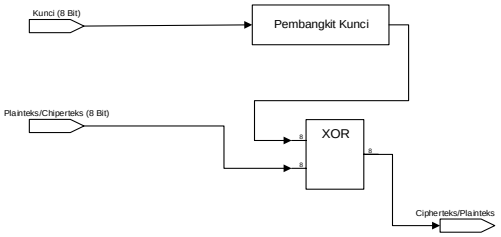
Gambar 11. Alur Kerja FSR 2



Gambar 12. Alur Kerja FSR 3

4.2.2 Perancangan Proses Enkripsi/Dekripsi

Tahap perancangan arsitektur proses enkripsi/dekripsi algoritma kriptografi vernam *cipher* dilakukan untuk memberikan gambaran proses algoritma kriptografi vernam *cipher* dalam bentuk rangkaian sistem digital. Proses enkripsi/dekripsi dimulai dengan dua buah *input*. *Input* yang pertama merupakan data yang akan dienkripsi/dekripsi, sedangkan *input* yang kedua merupakan kunci yang akan dibangkitkan menjadi *keystream*. Dalam penelitan ini proses enkripsi dan dekripsi menggunakan rancangan arsitektur yang sama. Dengan kata lain hanya terdapat satu rancangan untuk melakukan proses enkripsi dan dekripsi. *Input* pada proses enkripsi adalah *plainteks*, sedangkan yang menjadi *input* pada proses dekripsi ialah *cipherteks* yang merupakan hasil dari proses enkripsi yang telah dilakukan sebelumnya. Proses enkripsi dan dekripsi menggunakan kunci yang sama karena algoritma kriptografi vernam *cipher* termasuk ke dalam jenis algoritma kriptografi simetri. Arsitektur proses enkripsi/dekripsi algoritma kriptografi vernam *cipher* pada penelitan ini dapat dilihat pada Gambar 13.



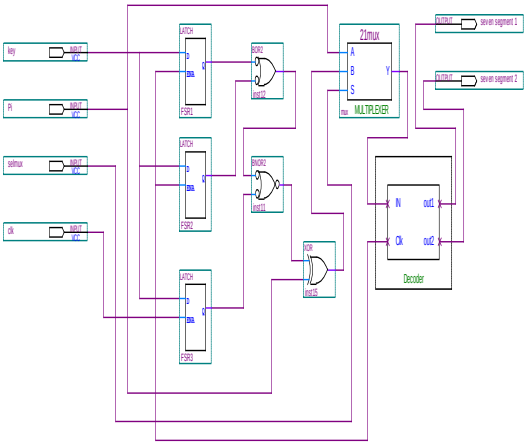
Gambar 13. Arsitektur Proses Enkripsi/Dekripsi Vernam *Cipher*

4.2.3 Perancangan Arsitektur Keseluruhan Sistem

Tahap terakhir dalam perancangan arsitektur algoritma kriptografi vernam *cipher* berbasis FPGA pada penelitan ini adalah perancangan arsitektur keseluruhan sistem. Tahap ini memberikan gambaran secara kesluruhan terhadap komponen penyusun sistem dalam bentuk skema rangkaian digital. Arsitektur

 ISSN : 2338-493X

keseluruhan sistem dalam penelitian ini merupakan gabungan dari arsitektur pembangkit kunci dan aristektur proses enkripsi/dekripsi serta komponen lain penyusun sistem secara keseluruhan. Arsistektur rancangan sistem dari impelementasi algoritma kriptografi vernam *cipher* berbasis FPGA dalam penelitian ini dapat dilihat seperti pada Gambar 14.



Gambar 14. Rancangan Sistem Secara Keseluruhan

4.3 Perancangan Kode VHDL Arsitektur Algoritma Vernam *Cipher*

Pada penelitian ini *entity* atau antarmuka sistem yang dirancang terdiri dari *input* dan *output*. Adapun rancangan *entity* pada impelementasi algoritma kriptografi vernam *cipher* berbasis FPGA ini dalam bahasa VHDL adalah sebagai berikut.

Kode Program 1. *Entitiy* Vcore

```
entity vcore is
port   (     pi_ci     : in std_logic_vector (7 downto 0);
             key       : in std_logic_vector (7 downto 0);
             Selmux    : in std_logic ;
             Clk       : in std_logic ;
             output1   : out std_logic_vector (7 downto 0);
             output2   : out std_logic_vector (7 downto 0)
          );
end vcore;
```

Architecture dalam VHDL mendeskripsikan implementasi rangkaian digital, dalam penelitian ini yaitu rangkaian digital sistem dari implementasi algoritma kriptografi vernam *cipher* berbasis FPGA. Adapun racanngan pernyataan *architecture* dalam VHDL pada penelitina ini adalah sebagai berikut.

Kode Program 2. *Architecutre* Vcore

```
architecture beav of vcore is
----deklarasi signal
--- deklarasi komponen register
begin
--- arsitektur keygenerator
--- arsitektur register or dan nor
--- proses display ke 7 segment
--- proses enkripsi dan dekripsi
```

4.3.1 Rancangan Kode VHDL Pembangkit Kunci

Racangan kode VHDL pembangkit kunci dari implentasi algoritma kriptografi vernam *cipher* berbasis FPGA pada penelitian ini adalah seperti berikut.

Kode Program 3. Pembangkti Kunci

```
Process
(pi,ci,clk,key,fsr1_out,fsr2_out,fsr3_out,regOR_out,r
egNOR_out)
begin
if clk ='1' then
fsr1_in <= (key(7) xor key (0))& key (7 downto 1);
fsr2_in <= (key(4) xor key (3))& key (7 downto 1);
fsr3_in <= (key(5) xor key (2))& key (7 downto 1);
regOR_in  <= fsr1_out or fsr2_out;
regNOR_in <= regOR_out nor fsr3_out
```

4.3.2 Rancangan Kode VHDL Proses Enkripsi/Dekripsi

Proses enkripsi/dekripsi dilakukan ketika mendapat *input* clk dengan nilai logika 1. Hasil enkripsi/dekripsi ditempatkan pada signal enkrip_dekrip yang merepresentasikan hasil dari proses enkripsi/dekripsi.

Kode Program 4. Proses Enkripsi/Dekripsi

```
---proses enkripsi dan dekripsi

process
(pi_ci,ci_pi,clk,key,fsr1_out,fsr2_out,fsr3_out,regOR
_out,regNOR_out,enkrip_dekrip)
begin
if clk ='1' then
fsr1_in <= (key(7) xor key (0))& key (7 downto 1);
fsr2_in <= (key(4) xor key (3))& key (7 downto 1);
fsr3_in <= (key(5) xor key (2))& key (7 downto 1);
regOR_in      <= fsr1_out or fsr2_out;
regNOR_in     <= regOR_out nor fsr3_out;
enkrip/dekrip  <= pi_ci xor regNOR_out;
ci_pi <= enkrip_dekrips;
```

 ISSN : 2338-493X

4.3.3 Rancangan Kode VHDL Decoder *Seven segment*

Rancangan kode VHDL *decoder seven segment* dari implementasi algoritma kriptografi vernam *cipher* berbasis FPGA dalam penelitian ini dapat dilihat seperti pada Kode Program 5 dan 6.

Kode Program 5. *Decoder Seven segment*1

```
process (outmux1)
begin
case outmux1 is
        when "0000" => output1 <= "00000011"; -- 0
        when "0001" => output1 <= "10011111"; -- 1
        when "0010" => output1 <= "00100101"; -- 2
        when "0011" => output1 <= "00001101"; -- 3
        when "0100" => output1 <= "10011001"; -- 4
        when "0101" => output1 <= "01001001"; -- 5
        when "0110" => output1 <= "01000001"; -- 6
        when "0111" => output1 <= "00011111"; -- 7
        when "1000" => output1 <= "00000001"; -- 8
        when "1001" => output1 <= "00001001"; -- 9
        when "1010" => output1 <= "00010001"; -- A
        when "1011" => output1 <= "11000001"; -- b
        when "1100" => output1 <= "01100011"; -- C
        when "1101" => output1 <= "10000101"; -- d
        when "1110" => output1 <= "01100001"; -- E
        when "1111" => output1 <= "01110001"; -- F
end case;
end process;
```

Kondisi atau nilai dari sinyal *output*1 dan *output*2 (sinyal *input* untuk *seven segment* 1 dan sinyal *input untuk seven segment* 2) tergantung pada kondisi atau nilai dari sinyal outmux1 dan sinyal outmux 2 yang dirujuk saat proses berlangsung.

Kode Program 6. *Decoder Seven segment2*

```
process (outmux2)
begin
case outmux2 is
        when "0000" => output2 <= "00000011"; -- 0
        when "0001" => output2 <= "10011111"; -- 1
        when "0010" => output2 <= "00100101"; -- 2
        when "0011" => output2 <= "00001101"; -- 3
        when "0100" => output2 <= "10011001"; -- 4
        when "0101" => output2 <= "01001001"; -- 5
        when "0110" => output2 <= "01000001"; -- 6
        when "0111" => output2 <= "00011111"; -- 7
        when "1000" => output2 <= "00000001"; -- 8
        when "1001" => output2 <= "00001001"; -- 9
        when "1010" => output2 <= "00010001"; -- A
        when "1011" => output2 <= "11000001"; -- b
        when "1100" => output2 <= "01100011"; -- C
        when "1101" => output2 <= "10000101"; -- d
        when "1110" => output2 <= "01100001"; -- E
        when "1111" => output2 <= "01110001"; -- F
end case;
end process;
```

4.3.4 Rancangan Kode VHDL Tampilan Pada *Seven segment*

Rancangan kode VHDL tampilan *seven segment* dari implementasi algoritma vernam *cipher* berbasis FPGA dalam penelitian ini dapat dilihat seperti pada kode program 7. Pada potongan kode VHDL tampilan *seven segment* tersebut terdapat dua *output signal* yaitu *output*1 dan *output*2. Kedua *output* ini merupakan masukan yang akan mengatur tampilan pada *seven segment*. *Seven segment* akan menampilkan data *plainteks* (pi) jika diberi nilai logika 0 pada *input* selmux dan menampilkan *cipherteks* (ci) jika *input* selmux bernilai logika 1.

Kode Program 7. Tampilan *Seven segment*

```
process (selmux,pi_ci,ci_pi)
begin
if selmux ='0' then
   outmux1 <= pi_ci (7 downto 4);
   outmux2 <= pi_ci (3 downto 0);
             else
   outmux1 <= ci_pi (7 downto 4);
   outmux2 <= ci_pi (3 downto 0);
 end if;
 end process;
```

4.4 Perancangan Keselurahan Sistem

Tahap terakhir pada tahapan peracangan adalah perancangan keseluruhan sistem. Pada tahap ini hasil rancangan dari implementasi algoritma vernam *cipher* berbasis FPGA dimasukan kedalam *chip* FPGA. Proses memasukan rancangan ke dalam FPGA dilakukan melalui proses *download* menggunakan perangkat tambahan yaitu Altera USB Blaster yang berfungsi menghubungkan antara komputer dengan FPGA (Altera UP2 *Education Kit*).

Gambar 15. Altera Usb Blaster

Sebelum melakukan proses *download* rancangan sistem ke dalam FPGA (Altera UP2 *Edukation Kit*), perlu dilakukan satu tahapan untuk menentukan pin-pin FPGA Altera Flex 10K yang akan digunakan. Proses menentukan pin (pin *assignment*) ini dilakukan agar FPGA nantinya dapat saling berkomunikasi dengan peralatan lain yang dibutuhkan dalam hal ini yaitu peralatan *input dan output*. Sinyal dari peralatan *input* dan *output* dikirim melalui pin-pin FPGA yang telah ditentukan.

Pada penelitian ini *input* sistem dimasukan dengan menggunakan *switch*. *Switch* pada Altera

 ISSN : 2338-493X

UP2 *Education Kit* akan bernilai 1 atau berlogika 1 jika dalam keadaan terbuka dan berlogika 0 jika dalam keadaan tertutup [2]. Konfigurasi Pin Altera Flek 10K yang digunakan dalam penelitian ini secara lengkap dapat dilihat pada tabel 3.

Tabel 3. Pin *Assignment*

No	Nama Signal	Jenis	Nama Pin	Piranti *Input/output*
1	clk	*Input*	PIN_120	*Switch*
2	key[7]	*Input*	PIN_109	*Switch*
3	key[4]	*Input*	PIN_110	*Switch*
4	key[3]	*Input*	PIN_111	*Switch*
5	key[4]	*Input*	PIN_113	*Switch*
6	key[3]	*Input*	PIN_114	*Switch*
7	key[2]	*Input*	PIN_115	*Switch*
8	key[1]	*Input*	PIN_116	*Switch*
9	key[0]	*Input*	PIN_117	*Switch*
10	*output*1[7]	*Output*	PIN_6	*Seven segment*
11	*output*1[4]	*Output*	PIN_7	*Seven segment*
12	*output*1[3]	*Output*	PIN_8	*Seven segment*
13	*output*1[4]	*Output*	PIN_9	*Seven segment*
14	*output*1[3]	*Output*	PIN_11	*Seven segment*
15	*output*1[2]	*Output*	PIN_12	*Seven segment*
16	*output*1[1]	*Output*	PIN_13	*Seven segment*
17	*output*1[0]	*Output*	PIN_14	*Seven segment*
18	*output*2[7]	*Output*	PIN_17	*Seven segment*
19	*output*2[4]	*Output*	PIN_18	*Seven segment*
20	*output*2[3]	*Output*	PIN_19	*Seven segment*
21	*output*2[4]	*Output*	PIN_20	*Seven segment*
22	*output*2[3]	*Output*	PIN_21	*Seven segment*
23	*output*2[2]	*Output*	PIN_23	*Seven segment*
24	*output*2[1]	*Output*	PIN_24	*Seven segment*
25	*output*2[0]	*Output*	PIN_25	*Seven segment*
26	pi_ci[7]	*Input*	PIN_41	*Switch*
27	pi_ci[4]	*Input*	PIN_40	*Switch*
28	pi_ci[3]	*Input*	PIN_39	*Switch*
29	pi_ci[4]	*Input*	PIN_38	*Switch*
30	pi_ci[3]	*Input*	PIN_36	*Switch*
31	pi_ci[2]	*Input*	PIN_35	*Switch*
32	clk	*Input*	PIN_120	*Switch*
33	pi_ci[0]	*Input*	PIN_33	*Switch*
34	Selmux	*Input*	PIN_126	*Switch*

5. PENGUJIAN DAN ANALISIS

5.1 Pengujian Rancangan Sistem

Pengujian rancangan sistem dilakukan untuk mengetahui apakah sistem sudah berjalan dengan baik dan sesuai dengan perancangan. Pengujian dilakukan terhadap komponen rancangan sistem yang terdiri dari pengujian rancangan pembangkit kunci, pengujian proses enkripsi/dekripsi, dan pengujian rancangan tampilan *seven segment*.

Pengujian dilakukan dengan menguji rancangan sistem melalui simulasi *functional* menggunakan *tools* waveform pada *software* Quartus II *Web edition*. Adapun tahapan yang dilakukan untuk melakukan pengujian rancangan sistem secara simulasi pada penelitian ini adalah sebagai berikut.

Pada penelitian ini dibuat sebuah tabel *plainteks* dan kunci yang digunakan untuk menguji implementasi sistem. Tabel ini terdiri dari empat belas karakter ascii yang digunakan sebagai *plainteks* dan dua buah karakter ascii yang digunakan sebagai kunci. Berikut ini merupakan *plainteks* dan kunci yang digunakan dalam pengujian tersebut.

Tabel 4. *Plainteks* Dan Kunci Dalam Pengujian

No	KARAKER /ASCII	KONVERSI		KETERANGAN
		HEX	BINER	
1	#	23	00100011	*Plainteks*
2	S	53	01010011	*Plainteks*
3	I	69	01101001	*Plainteks*
4	s	73	01110011	*Plainteks*
5	K	4B	01001011	*Plainteks*
6	o	6F	01101111	*Plainteks*
7	m	6D	01101101	*Plainteks*
8	0	30	00110000	*Plainteks*
9	8	38	00111000	*Plainteks*
10	(	28	00101000	*Plainteks*
11	O	4F	01001111	*Plainteks*
12	k	6B	01101011	*Plainteks*
13	!	21	00100001	*Plainteks*
14	)	29	00101001	*Plainteks*
15	.	2E	00101110	Kunci
16	@	40	0100 0000	Kunci

5.1.1 Pengujian Rancangan Pembangkit Kunci

Pengujian rancangan pembangkit kunci dilakukan untuk melihat apakah rancangan pembangkit kunci sudah berjalan dengan baik dan sesuai dengan perancangan awal yang diinginkan. Pengujian dilakukan melalui simulasi.

Pada pengujian ini masing-masing kunci (key) karakter ascii "." (dot) dengan kode biner 00101110 dan karakter ascii "@" dengan kode biner 01000000 dibangkitkan dengan memberikan masukan *input* clk (logika 1). Ketika clk bernilai 1 proses membangkitkan

 ISSN : 2338-493X

kunci dilakukan sehingga menghasilkan sebuah *keystream*. Dalam proses ini *keystream* yang dihasilakn adalah kode biner 01101000 untuk kunci dengan karakter "." (dot) dan keystream dengan kode biner 11011111 untuk kunci dengan karakter "@".

5.1.2 Pengujian Rancangan Proses Enkripsi/Dekripsi

Pengujian rancangan proses enrkipsi/dekripsi dilakukan untuk melihat apakah rancangan proses enkripsi/dekripsi sudah berjalan dengan baik dan sesuai dengan perancangan awal yang diinginkan. Pengujian rancangan sistem terhadap proses enkripsi dilakukan sebanyak dua puluh delapan kali dimana masing-masing karakter ascii plainteks dilakukan pengujian proses enkripsi sebanyak dua kali yaitu dengan menggunakan dua buah kunci yang berbeda. Adapun kunci yang digunakan dalam proses ini yaitu karakter ascii "." dan karakter ascii "@". Pengujian ini dilakukan melalui simulasi pada *tools* waveform dengan membeikan *input* berupa kunci (*key*), data plainteks (pi), dan clk dengan nilai logika satu. Dari hasil pengujian yang telah dilakukan, rancangan proses enkripsi dapat berjalan dengan baik dan sesuai dengan perancangan.

Tahap Pengujian berikutnya adalah melakukan pengujian rancangan sistem terhadap proses dekripsi. Pada pengujian ini data masukan adalah *cipherteks* hasil dari pengujian rancangan sistem pada proses enrkipsi yang telah dilakukan sebelumnya.

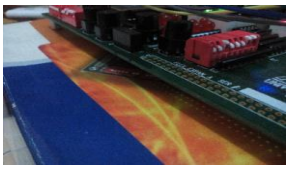
5.1.3 Pengujian Rancangan Tampilan *Sevent Segment*

Pengujian rancangan tampilan *seven segment* dari implementasi algoritma kriptografi vernam *cipher* berbasis FPGA pada penelitan ini dilakukan untuk melihat apakah rancangan tampilan *seven segment* sudah berjalan dengan baik dan sesuai dengan perancangan awal yang diinginkan. Tampilan *seven segment* pada penelitian ini adalah hasil *decoder* dari dua kelompok bit *cipherteks*/*plainteks* yang merupakan hasil dari proses enkripsi/dekripsi. Kelompok bit pertama (bit7-bit4 *plainteks*/*cipherteks*) berada pada *signal output*1 dan kelompok bit kedua (bit 3– bit 0 *plainteks*/*cipherteks*) berada pada *signal output*2.

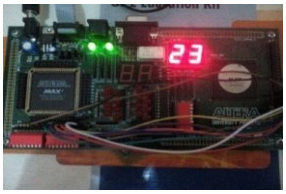
5.2 Pengujian Implementasi Sistem Algoritma Vernam *Cipher* Pada FPGA

Tahap pengujian implementasi sistem pada FPGA merupakan tahap terakhir dalam proses pengujian dari implementasi algoritma vernam *cipher* berbasis FPGA pada penelitian ini. Proses pengujian implementasi sistem bertujuan untuk melihat hasil rancangan implementasi sistem secara keseluruhan. Pengujian implementasi sistem dilakukan pada proses enkripsi dan dekripsi. Pada proses enkripsi, data *input* adalah *plainteks*, kunci, *input* proses enkrpsi/dekripsi (clk), serta *input* tampilan *seven segment* (selmux). Sedangkan pada proses dekripsi data *input* berupa *cipherteks* hasil enkripsi dari proses enkripsi sebelumnya, kunci yang sama seperti pada proses enkripsi, *input* proses enkrpsi/dekripsi (clk), serta *input* tampilan *seven segment* (selmux).

Pengujian pertama pada implementasi sistem terhadap proses enkripsi dilakukan dengan memasukan data *plainteks* berupa karakter "#" dengan bilangan biner 0010 0011. *Input* selanjutnya yang dimasukan adalah *input* tampilan *seven segment* (selmux) pada *switch* A (7) dan clk pada *switch* A (8). *Input* terakhir yang dimasukan adalah *input* kunci pada *switch* B (1 sampai 8). Pada kasus ini kunci yang dimasukan adalah karakter "." dengan kode biner 0010 1110.

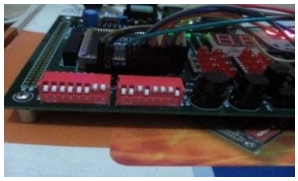


Gambar 16. Input Data *Plainteks* Untuk Karakterk Ascii "#" Pada *Switch*

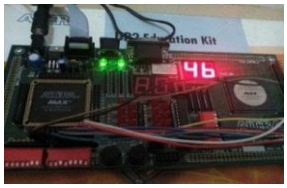


Gambar 17. Tampilan *Seven segment* Saat *Plainteks* Karakter # di Input

 ISSN : 2338-493X



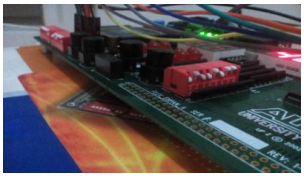
Gambar 18. Input Kondisi Enkripsi dan Tampilan *Seven Segmen* serta Kunci Pada Switch



Gambar 19. Tampilan *Seven segment* Saat Proses Enkripsi Dilakukan

Pengujian impelementasi sistem dilakukan pada empat belas data *plainteks* dan dua kunci seperti yang terdapat pada tabel 4. Pengujian ini dilakukan dengan menguji satu persatu kondisi *input* dan melihat hasil *output* yang terjadi.
Setelah melakukan pengujian implementsi sistem pada proses enkripsi, selanjutnya dilakukan pengujian implementasi sistem pada proses dekripsi. Pengujian pada proses dekripsi dilakukan sama seperti pengujian implementasi sistem pada proses enkripsi. Pengujian implementasi sistem terhadap proses dekripsi dilakukan dengan memasukan data *chiperteks* hasil enkripsi dari tahap pengujian yang telah dilakukan sebelumnya.

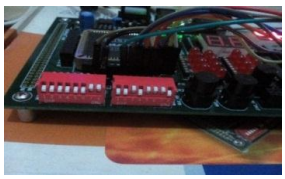
Pengujian pertama dilakukan dengan memasukan *cipherteks* berupa karakter "K" dengan kode biner 0100 1011 yang merupakan hasil enkripsi dari *plainteks* berupa karakter "#". Gambar 20 dan 21 berikut ini memperlihatkan kondisi *switch* dan tampilan *seven segment* ketika *input cipherteks* berupa karakter "K" dimasukan



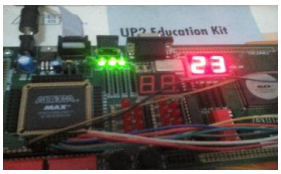
Gambar 20. Input Data *Cipherteks* Untuk Karakterk Ascii "K" Pada *Switch*

Gambar 21. Tampilan *Seven segment* Saat *Cipherteks* Karakter "K" di Input

Selanjutnya proses yang dilakukan adalah memberikan input kunci yang sama seperti pada proses enkripsi di tahap pengujian sebelumnya yaitu kunci berupa karakter "." dengan kode biner 0010 1110. Kemudian memasukan *input* proses dekripsi (clk) dengan nilai satu dan menampilkan hasilnya pada *seven segment* dengan memberikan *input* tampilan *seven segment* (selmux) dengan nilai satu. Gambar 22 dan 23 berikut ini memperlihatkan kondisi *switch* dan tampilan *seven segment* saat proses dekripsi dilakukan.



Gambar 22. Input Kondisi Dekripsi dan Tampilan *Seven Segment* serta Kunci Pada Switch



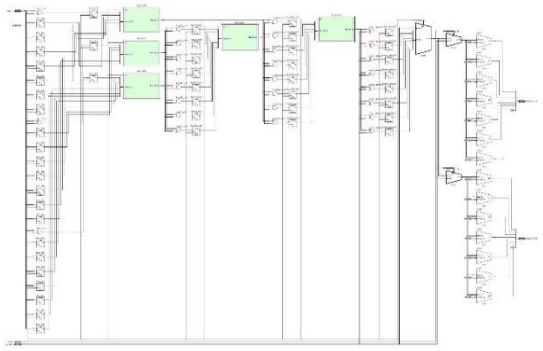
Gambar 23. Tampilan *Seven segment* Saat Proses Dekripsi Dilakukan

Pengujian implementasi sistem pada proses dekripsi dilakukan pada dua puluh delapan *cipherteks* hasil proses enkripsi yang telah dilakukan sebelumnya. Pengujian ini dilakukan dengan munguji satu persatu kondisi *input* dan melihat hasil *output* yang terjadi. dekripsi dapat dilihat pada lampiran dua dalam penulisan tugas akhir ini.

5.2 Hasil Implementasi

Dari keseluruhan hasil pengujian yang telah dilakukan, implementasi algoritma vernam *cipher* berbasis FPGA ini dapat berfungsi sesuai

dengan perancangan. Sistem ini dapat melakukan proses enkripsi dan dekripsi yang sesuai yaitu hasil enkripsi (*cipherteks*) setelah di dekripsi menghasilkan hasil yang sama dengan *plainteks* awal. Selain itu *display seven segment* sebagai tampilan *plainteks* dan *cipherteks* juga dapat berjalan dengan baik. Gambar 24 Berikut ini memperilhatkan hasil implementasi algoritma vernam *cipher* ke dalam FPGA pada penelitian ini.



Gambar 24. RTL Implementasi Algoritma Kriptografi Vernam *Cipher* Pada FPGA

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan penelitian dan pengujian yang telah dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Dari penelitian yang telah dilakukan dihasilkan arsitektur *hardware* algoritma kriptografi vernam *cipher* berbasis FPGA.
2. Arsitektur *hardware* algoritma kriptografi vernam *cipher* berhasil diterjemahkan kedalam bahasa VHDL.
3. Implementasi yang dihasilkan merupakan sistem yang mampu melakukan proses enrkipsi dan dekripsi data dengan lebar data 8 bit.
4. Proses enkripsi dan dekripsi dilakukan dengan menggunakan implementasi sistem dari rancangan sistem yang sama.
5. Jumlah Pin Altera Flex 10K yang terpakai dalam implementasi ini adalah 34 Pin atau 18% dari total Pin Altera Flex 10K.
6. Jumlah Elemen Logika Altera Flex 10K yang terpakai dalam implementasi ini adalah sebanyak 76 elemen logika atau 2 % dari total elemen logika yang dimiliki oleh Altera Flex 10K.
7. Pin Pada Altera Flex 10K dapat dikoneksikan dengan input maupun output tambahan dengan terlebih dahulu melakukan konfigurasi pada Pin Altera Flex 10K dan Flex Expan yang terdapat pada Altera UP2 *Education Kit*.
8. Pada Penelitian ini pengujian secara simulasi dan pengujian terhadap implementasi sistem secara langsung pada FPGA menunjukkan hasil yang sama.

6.2 Saran

Dalam penelitian ini, implementasi yang telah dilakukan secara fungsi dapat bekerja dengan baik, namun masih memerlukan pengembangan lebih lanjut agar implementasi dapat diaplikasikan dalam bentuk yang lebih nyata. Saran-saran untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Mengembangkan sistem agar dapat melakukan proses enkripsi dan dekripsi data dengan lebar data yang lebih besar.
2. Memperhitungakan dan menganalisa *timing* pada sistem saat melakukan proses enkripsi/dekripsi ketika di aplikasikan pada data yang lebih besar.
3. Melakukan analisa lebih jauh terhadap keamanan yang dihasilakan dari sistem yang dibuat.

DAFTAR PUSTAKA

[1] Abdurohman, M. (2014). Organisasi & Arsitektur Komputer. Bandung: Informatika Bandung.

[2] Altera Cooperation. (2004). University Program UP2 Education Kit User Guide. San Jose.

[3] Jatmiko, W., & dkk. (2011). Implementasi Berbagai Algoritma Neural Network Dan Wavelet Pada Field Programmable Gate Array. Jakarta: Fakultas Ilmu Komputer Universitas Indonesia.

[4] Kromodimoeljo, S. (2009). Teori dan Aplikasi Kriptografi. Jakarta: SPK IT Consulting.

[5] Schneier, B. (1996). Applied Cryptography Second Edition. New York: John Wiley & Son.

[6] Sholeh, M., & Hamokwarong, J. V. (2011). Aplikasi Kriptografi Dengan Metode Vernam *Cipher* Dan Metode Permutasi Biner. Momentum, Vol. 7, No. 2, 8-13.