UNIVERSITAS INDONESIA

PERANCANGAN DAN IMPLEMENTASI ALGORITMA ENKRIPSI ARCFOUR PADA PERANGKAT KRIPTOGRAFI BERBASIS FPGA

SKRIPSI

MOHAMAD SYAHRAL
0607199666

FAKULTAS TEKNIK ELEKTRO
DEPOK
2011

UNIVERSITAS INDONESIA

PERANCANGAN DAN IMPLEMENTASI ALGORITMA ENKRIPSI ARCFOUR PADA PERANGKAT KRIPTOGRAFI BERBASIS FPGA

SKRIPSI

Diajukan sebagai syarat untuk memperoleh gelar Sarjana Teknik

MOHAMAD SYAHRAL
0607199666

FAKULTAS TEKNIK ELEKTRO
DEPOK
2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	: MOHAMAD SYAHRAL
NPM	: 0706199666
Tanda Tangan	: .....
Tanggal	: 6 Januari 2011

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	MOHAMAD SYAHRAL
NPM	:	0706199666
Program Studi	:	TEKNIK ELEKTRO
Judul Skripsi	:	PERANCANGAN DAN IMPLEMENTASI ALGORITMA ENKRIPSI ARCFOUR PADA PERANGKAT KRIPTOGRAFI BERBASIS FPGA

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Ir. Wahidin Wahab, Msc, Ph.D ( )

Penguji : Abdul Muis, ST, M.Eng, Ph.D ( )

Penguji : Dr. Ir. Ridwan Gunawan, MT ( )

Ditetapkan di : Depok

Tanggal : 06 Januari 2011

KATA PENGANTAR

Alhamdulillahi Robbil 'alamiin, segala puji dan syukur kehadirat Allah SWT yang telah terus membangkitan semangat penulis, dan atas berkat dan rahmatNya-lah penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan sebagai salah satu syarat untuk mendapatkan gelar Sarjana Teknik pada Fakultas Teknik Elektro Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada :

1. Ir, Wahidin Wahab, M.Sc, Ph.D, selaku dosen pembimbing yang telah menyediakan waktu, tenaga dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini, serta kesabaran ekstra dalam menghadapi segala kekurangan saya.
2. Sumber inspirasi dan pelipur lara saya, istri tercinta Santi Rahayu dan ketiga anak-anak saya yang selalu ceria, Dhuha, Ilham dan Mila.
3. Orang tua dan segenap saudara-saudara saya yang tak pernah putus mendo'akan dan menyemangati saya.
4. Rekan-rekan staf bidang Perangkat Keras Lembaga Sandi Negara yang telah banyak membantu dan menyemangati saya terus menerus.
5. Adik-adik Mahasiswa STSN Tingkat IV Rancang Bangun yang selalu memberikan suasana baru dan ide-ide baru dalam mengembangkan teknologi FPGA.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 6 Januari 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: MOHAMAD SYAHRAL
NPM	: 0706199666
Program Studi	: Teknik Elektro
Departemen	: Teknik
Fakultas	: Teknik
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right) atas karya ilmiah saya yang berjudul :

Perancangan Dan Implementasi Algoritma Enkripsi Arcfour Pada Perangkat Kriptografi Berbasis FPGA

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 6 Januari 2011
Yang menyatakan

(MOHAMAD SYAHRAL)

ABSTRAK

Nama : Mohamad Syahral
Program Studi : Teknik Elektro
Judul : Perancangan dan Implementasi Algoritma Enkripsi Arcfour Pada Perangkat Kriptografi Berbasis FPGA .

Kriptografi merupakan salah satu metode yang dapat digunakan untuk mengamankan informasi karena kriptografi mengkodekan suatu informasi sedemikian rupa sehingga informasi tersebut tidak dapat diketahui oleh pihak-pihak yang tidak berhak mengetahuinya.

Dalam tugas akhir ini dibahas tentang perancangan dan inplementasi suatu algoritma enkripsi dalam bentuk modul perangkat keras yang dapat dihubungkan langsung ke komputer dan mengenkripsi data-data yang akan dikirimkan ke komputer lainnya melalui sebuah modem yang terhubung ke PSTN. Algoritma yang digunakan adalah algoritma arcfour dan diimplementasikan ke FPGA Xilinx Spartan-IIE LC Development Board dengan menggunakan bahasa deskripsi perangkat keras VHDL.

Pada tugas akhir ini algoritma enkripsi arcfour berhasil diimplementasikan ke FPGA Xilinx Spartan-IIE LC Development Board dan melakukan enkripsi data-data berupa teks yang dikirimkan serta mendekripsi data yang diterima. Sedangkan untuk data berupa *file,* proses enkripsi dan dekripsi masih belum berhasil dengan sempurna.

Kata kunci :
Algoritma enkripsi *Arcfour,* FPGA Xilinx Spartan-IIE LC Development Board, Bahasa Deskripsi Perangkat Keras VHDL.

ABSTRACT

Name : Mohamad Syahral
Study Program : Electrical and Electronic Engineering
Title : Design and Implementation of Arcfour encryption algorithm Into FPGA Based Cryptographic Device .

Cryptography is one of the method that use to secure information for transmission, where cryptography encodes the information in certain way so that the information would not be known by anyone that have no access to the information.

The focus of this study is designing and implementation one of encryption algorithm in a hardware device that can be connected directly to computer and decode the data that can be transmitted to another computer through a modem that connected to PSTN. Encryption algorithm that used is arcfour algorithm and implemented in Xilinx Spartan-IIE LC Development Board using VHDL Hardware Description Languange.

In this study, the arcfour algorithm has been successfully implemented in FPGA Xilinx Spartan-IIE LC Development Board, encrypting text data that transmitted and decrypting received text data. But for data transfer of files, encryption and decryption process was not yet perfect successful.

Keyword :
Arcfour encryption algorithm , FPGA Xilinx Spartan-IIE LC Development Board, VHDL Hardware Description Language.

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PERNYATAAN ORISINALITAS ii
LEMBAR PENGESAHAN iii
KATA PENGANTAR iv
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH v
ABSTRAK vi
DAFTAR ISI viii
DAFTAR GAMBAR x
DAFTAR TABEL xi
1. PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Pembatasan Masalah 2
1.4 Metode Perancangan 3
1.5 Tujuan dan Manfaat 3
1.6 Sistematika Penulisan 4

2. LANDASAN TEORI 6
2.1 Algoritma Enkripsi Arcfour 6
2.1.1 Key-Schedulling Algorithm (KSA) 6
2.1.2 Pseudo-Random Generation Algorithm (PRGA) 7
2.2 Field Programmable Gate Array (FPGA) 8
2.2.1 Pengertian FPGA 8
2.2.2 Arsitektur FPGA 9
2.2.3 Mengkonfigurasikan FPGA 10
2.2.4 Spesifikasi FPGA Xilinx Spartan-IIE LC Development Board 12
2.3 VHSIC Hardware Description Language (VHDL) 13
2.3.1 Pengertian 13
2.3.2 Struktur Dasar 14
2.3.3 Metode Verifikasi 15
2.3.4 Teknik Deskripsi VHDL 16
2.4 Metode Perancangan 17
2.4.1 Waterfall ASIC Design Flow 17
2.4.2 Perancangan Sistem dengan VHDL 18
2.4.3 Data Flow Diagram (DFD) 21

3. PERANCANGAN DAN IMPLEMENTASI SISTEM 27
3.1 Deskripsi dan Spesifikasi Sistem 27
3.1.1 Deskripsi Sistem 27
3.1.2 Spesifikasi Sistem 28
3.2 Pemodelan Sistem 28
3.2.1 Diagram Konteks 28
3.2.2 DFD Level Nol 29

3.2.3 DFD Level Satu Modul Enkrip …….. 29
3.2.4 DFD Level Satu Modul Dekrip ……. 29
3.3 Perancangan Arsitektur Sistem ……. 30
3.4 Perancangan Sistem Dengan VHDL ……..……. 30
3.4.1 Rancangan Modul VHDL UART_Tx……. ... 30
3.4.2 Rancangan Modul VHDL UART_Rx …… 34
3.4.3 Rancangan Modul VHDL Arcfour Cryptoprocessor ……. 38
3.5 Implementasi Rancangan …….. 48
3.5.1 Skema Rancangan …….. 48
3.5.2 HDL Synthesis ……. 49
3.5.3 Advance HDL Synthesis ……. 49
3.5.4 Low Level Synthesis ……. 50
3.5.5 Rangkuman Synthesis …….. 51

4. PENGUJIAN DAN ANALISIS …… 52
4.1 Pengujian ……..…….. 52
4.1.1 Verifikasi Formal……. ……... 52
4.1.2 Ujicoba Operasional Kirim Terima File …… 53
4.2 Analisis…….…… 53
4.2.1 Rekonstruksi Kode VHDL……..…… 53
4.2.2 Simulasi Fungsional …….. 53
4.2.3 Implementasi Rancangan …….. 53
4.2.4 Verifikasi Formal ……..……... 54
4.2.5 Ujicoba Operasional Kirim Terima File …… 54

5. KESIMPULAN …….. 55

DAFTAR REFERENSI …….. 56

DAFTAR GAMBAR

Gambar 2.1 Sebuah Logic Cell pada FPGA ……. 8
Gambar 2.2 Programmable Interconnects ……. 9
Gambar 2.3 Arsitektur Dasar FPGA…….…… 10
Gambar 2.4 Arsitektur FPGA Spartan-II series …….. 10
Gambar 2.5 Xilinx FPGA Spartan-IIE LC Development Board …….. 13
Gambar 2.6 Blok Diagram IIE LC Development Board …….. 13
Gambar 2.7 Struktur Dasar VHDL …….. 15
Gambar 2.8 Waterfall ASIC Design Flow …….. 17
Gambar 2.9 Diagram Alir Perancangan Sistem dengan VHDL…….. 19
Gambar 2.10 Komponen DFD ……. 22
Gambar 3.1 Skema Rancangan…….…… 28
Gambar 3.2 Conteks Diagram Modul Kripto …….. 28
Gambar 3.3 DFD Level Nol ……..……... 29
Gambar 3.4 DFD Level Satu Modul Enkrip ……. 29
Gambar 3.5 DFD Level Satu Modul Dekrip ……. 29
Gambar 3.6 Arsitektur Modul Enkripsi ……. 30
Gambar 3.7 UART_Tx State Diagram…… 30
Gambar 3.8. UART_Rx State Diagram …….. 34
Gambar 3.9 Simulasi Fungsional 0 s/d 25.000 ns …….. 46
Gambar 3.10 Simulasi Fungsional 25.000 s/d 50.000 ns …….. 47
Gambar 3.11 Simulasi Fungsional 50.000s/d 75.000 ns …….. 47
Gambar 4.1 Hasil Pengujian Rangkaian Kunci Arcfour …….. 51

DAFTAR TABEL

Tabel 3.1	HDL Synthesis Report ……..…….	48
Tabel 3.2	Advance HDL Synthesis Report ……..	48
Tabel 3.3	HDL Low Level Synthesis Report …….. …….	50
Tabel 3.4	Design Summary Report ……..	50
Tabel 3.5	Hasil Ujicoba Kirim Terima File ……..…….	54

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang sangat cepat dan derasnya kebutuhan masyarakat terhadap komunikasi merupakan fenomena yang tengah terjadi pada awal abad 21 ini. Fenomena ini kemudian dimanfaatkan dengan baik oleh kalangan dunia usaha sehingga tersedianya begitu banyak layanan komunikasi publik yang dapat dinikmati oleh masyarakat, terutama dengan pesatnya perkembangan teknologi komunikasi seluler atau nirkabel, baik komunikasi suara maupun komunikasi data.

Seiring dengan kebutuhan terhadap komunikasi tanpa disadari sebenarnya masyarakat juga memiliki kebutuhan terhadap keamanan informasi yang dimilikinya, bahkan kebutuhan akan keamanan informasi tersebut sebenarnya berada didepan sebelum kebutuhan terhadap komunikasi. Sebagai contoh, jika seseorang ingin menyampaikan suatu informasi penting dan rahasia kepada orang lain, maka orang tersebut lebih memilih bertatap muka dibandingkan menggunakan layanan komunikasi publik. Kemudian bagaimana halnya dengan komunikasi di kalangan pemerintahan atau militer, sebagian besar informasi yang akan dikomunikasikan memiliki tingkat kerahasiaan tertentu yang akan dikirimkan ke seluruh jajarannya.

Pada layanan komunikasi publik sangat rentan terhadap ancaman dari pihak ketiga untuk dapat menyadap, merusak dan mengubah informasi yang dikirimkan, dengan demikian dibutuhkan suatu layanan keamanan informasi dengan menggunakan suatu metode tertentu yang dapat digunakan untuk mengamankan informasi dari pihak-pihak yang tidak berhak untuk mengetahuinya.

Kriptografi merupakan salah satu metode yang dapat digunakan untuk mengamankan informasi karena kriptografi mengkodekan suatu informasi sedemikian rupa sehingga informasi tersebut tidak dapat diketahui oleh

pihak-pihak yang tidak berhak mengetahuinya, selain menjamin kerahasiaan informasi (*confidentiality)*, teknik kriptografi juga dapat menjamin keutuhan data (*data integrity)*, otentikasi *(authentication)* dan tahan terhadap penyangkalan *(non repudation).*

Kriptografi dapat diterapkan dengan mengimplementasikan suatu algoritma kriptografi baik dalam bentuk perangkat lunak maupun perangkat keras. Implementasi pada perangkat lunak dapat berupa aplikasi pada personal komputer, telepon genggam maupun perangkat komunikasi lainnya, sedangkan implementasi pada perangkat keras dapat dilakukan dengan berbasis Mikroprosessor, Mikrokontroler, Digital Signal Processing (DSP) maupun Programmable Logic Device (PLD) seperti Field Programmable Gate Array (FPGA) atau Complex Programmable Logic Device (CPLD).

Implementasi algoritma kriptografi dalam bentuk perangkat lunak sudah banyak dijumpai pada layanan-layanan yang disediakan oleh kalangan dunia usaha maupun perbankan, seperti transaksi *online*, sms banking dan internet banking, sedangkan implementasi algoritma kriptografi pada perangkat keras yang dapat digunakan untuk mengamankan komunikasi layanan publik masih sedikit dijumpai, untuk itu pada penulisan tugas akhir ini menitikberatkan pada implementasi algoritma kriptografi pada perangkat keras untuk menjamin kerahasiaan informasi yang akan dikomunikasikan.

1.2 Rumusan Masalah

Bagaimana mengimplementasikan algoritma kriptografi ke dalam suatu perangkat keras agar dapat mengamankan komunikasi atau perpindahan data dari suatu komputer ke komputer lainnya melalui suatu saluran komunikasi dengan proses enkripsi.

1.3 Pembatasan Masalah

Dalam tugas akhir ini rumusan masalah diatas akan dikhususkan lagi sesuai dengan batasan-batasan sebagai berikut :

a. Algoritma kriptografi yang akan diimplementasikan adalah algoritma enkripsi *Arcfour* dengan rangkaian kunci tetap.
b. Perangkat keras yang digunakan adalah Field Programmable Gate Array (FPGA) Xilinx Spartan-IIE LC Development Board.
c. Bahasa HDL yang digunakan adalah Very High speed IC Hardware Description Laguange (VHDL).
d. Saluran komunikasi data pada tugas akhir ini adalah saluran komunikasi yang melalui perangkat modem dan PABX, modem yang digunakan adalah Radicom Modemslim3.
e. Data yang akan dikirimkan berupa text dan *file transfer* melalui aplikasi Microsoft Hyper Terminal.
f. Sistem komunikasi diasumsikan digunakan pada kondisi ideal tanpa ada gangguan di saluran transmisi, sehingga perangkat yang akan dikembangkan tidak menggunakan protokol komunikasi tertentu.

1.4 Metode Perancangan

Perancangan sistem di dalam tugas akhir ini mengadaptasi beberapa metode perancangan, yakni ;

a. Waterfall ASIC Design Flow
b. Perancangan sistem dengan VHDL
c. Teknik pemodelan *Data Flow Diagram* (DFD)
d. Simulasi fungsional dengan menggunakan *VHDL Testbench Code*
e. Menggunakan aplikasi Xilinx Webpack ISE 8.2i untuk melakukan synthesis hingga implementasi ke dalam FPGA Xilinx development board

1.5 Tujuan dan Manfaat

Tujuan :

a. Merancangbangun suatu perangkat kriptografi yang mampu menjamin keamanan informasi pada suatu sistem komunikasi.
b. Mengimplementasikan algoritma enkripsi *Arcfour* pada FPGA Xilinx Development Board dengan menggunakan bahasa pemrograman VHDL.

c. Memahami alur proses perancangan perangkat kriptografi berbasis FPGA.
d. Sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Teknik pada Program Studi Elektro Fakultas Teknik Universitas Indonesia.

Manfaat :

a. Menjadi langkah awal dalam proses rancang bangun sistem komunikasi data atau suara yang aman.
b. Diharapkan mampu diintegrasikan pada suatu sistem komunikasi lainnya untuk menjamin keamanan informasi.
c. Diharapkan dapat memperkaya khazanah penelitian tentang implementasi algoritma kriptografi berbasis Field Programmable Gate Array (FPGA).
d. Menambah wawasan dan meningkatkan keahlian dalam perancangan sistem on chip (IC) atau Cryptoprocessor dengan bahasa pemrograman VHDL.

1.6 Sistematika Penulisan

Bab pertama berisi latar belakang, rumusan masalah, pembatasan masalah, metode penelitian, tujuan dan manfaat penulisan serta sistematika penulisan tugas akhir.

Bab kedua berisi tentang teori – teori yang mendasari penulisan tugas akhir, antara lain, algoritma enkripsi *Arcfour*, FPGA Xilinx Development Board, bahasa deskripsi perangkat keras VHDL, Radicom Modemslim3, metode-metode perancangan yang digunakan dan segala sesuatu yang berkaitan dengan masalah penelitian.

Bab ketiga membahas tentang perancangan dan implementasi pengamanan komunikasi berbasis FPGA menggunakan algoritma enkripsi *Arcfour* dan menjelaskan proses dan hasil implementasi ke dalam FPGA Xilinx Development Board.

Bab keempat menjelaskan mengenai pengujian dan analisis terhadap hasil implementasi pengamanan komunikasi berbasis FPGA.

Bab kelima berisi kesimpulan tentang proses perancangan hingga pengujian algoritma kriptografi enkripsi *Arcfour* berbasis FPGA.

BAB 2
LANDASAN TEORI

2.1 Algoritma Enkripsi *Arcfour*

Algoritma enkripsi *Arcfour* merupakan algoritma enkripsi yang diyakini sesuai dengan algoritma enkripsi RC4 yang dikembangkan oleh Ron Rivest dari RSA Security pada tahun 1987 yang selama 7 tahun menjadi properti dari RSA.

Kemudian pada bulan September 1994 deskripsi tentang RC4 dikirimkan ke *Cypherpunks mailing list* sebagai surat kaleng, lalu dirimkan lagi ke *sci.crypt newsgroup* dan dari sana tersebar luar ke berbagai situs di internet. Algoritma yang tersebar ini telah dikonfirmasi sebagaimana telah ditelaah bahwa rangkaian kunci yang dihasilkan cocok dengan aplikasi yang menggunakan algoritma RC4 yang memiliki lisensi, yang kemudian algoritma enkripsi tersebut dikenal dengan *Alleged* RC4 atau Arcfour.

RC4 sendiri merupakan algoritma enkripsi yang sering digunakan pada protokol-protokol yang cukup terkenal seperti *Secure Sockets Layer* (SSL) untuk melindungi lalu lintas di internet dan WEP pada jaringan nirkabel. RC4 memiliki keunggulan khususnya pada kederhanaan rancangannya dan kecepatannya pada perangkat lunak, dan dapat digunakan dengan variasi panjang kuncinya, mulai dari 0 hingga 256 byte.

Implementasi algoritma enkripsi Arcfour membutuhkan rangkaian kunci dengan panjang 0 hingga 256 bytes sesuai dengan yang diinginkan dan sebuah larik dengan ukuran 256 x 8 bit sebagai S-Box sebut saja larik "S". Proses enkripsi dilakukan dalam 2 (dua) fase, yakni :

2.1.1 Key-Schedulling Algorithm (KSA)

Tahapan-tahapan pada KSA adalah sebagai berikut :

- Menginisialisasi larik S (S-Box) dengan nilai yang urutnya masing-masing yakni dari 0 sampai 255.

  S[0] = 0; S[1] = 1; S[2] = 2; S[3] = 3; ……S[255] = 255

- Melengkapi rangkaian kunci menjadi larik dengan ukuran 256 bytes dengan melakukan pengulangan rangkaian kunci jika kunci yang digunakan kurang dari 256 bytes, sebut saja larik S2.
- Melakukan permutasi S-Box dengan menggunakan dua pointer i dan j, selanjutnya menukar isi S-Box yang ke-i dengan isi S-Box yang ke-j, sebanyak 256 putaran, dengan nilai i bergerak dari 0 hingga 255, sedangkan nilai merupakan hasil penjumlahan modulo 256 antara isi S-Box yang ke-i, S-Box yang ke-j dan nilai j pada putaran sebelumnya, nilai j diberi nilai awal 0.

  for (i = 0; i < 256; i = i + 1)

  {j = (j + S [i] + S2 [i]) mod 256;

  temp = S [i];

  S [i] = S [j];

  S [j] = temp;}
- Inisialisasi kembali nilai i dan j menjadi 0

2.1.2 Pseudo-Random Generation Algorithm (PRGA)

Kunci acak yang akan digunakan untuk melakukan proses enkripsi dihasilkan oleh PRGA, banyaknya kunci acak yang dihasilkan sesuai dengan banyaknya data yang akan dienkripsi, data dan kunci acak yang dienkripsi sebanyak 1 bytes setiap kali penyandian dengan menggunakan operasi XOR.

Proses pada PRGA hampir sama dengan proses permutasi pada KSA, perbedaannya adalah perhitungan nilai j tidak melibatkan larik S2, prosesnya tidak dibatasi sebanyak 256 putaran tetapi sesuai banyaknya data yang akan dienkripsi.

Kunci acak (sebut saja K) yang dihasilkan diperoleh dari isi S-Box yang ke-t, t adalah hasil penjumlahan modulo 256 dari isi S-Box yang ke-i dan isi S-Box yang ke-j.

i = (i+1) % 256;

j = (j + S[i]) % 256;

temp = S [i];

S [i] = S [j];

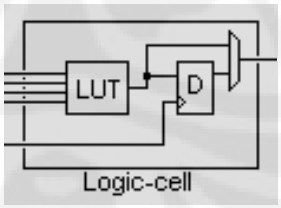
```
S [j] = temp;
t = (S [i] + S [j]) % 256;
K = S [t];
```

2.2 Field Programmable Gate Array (FPGA)

2.2.1 Pengertian FPGA

Field Programmable Gate Array (FPGA) merupakan komponen silicon seperti halnya *Integrated Circuit* (IC) yang fungsi dan rancangannya dapat dikonfigurasi oleh pengguna atau perancang dengan menggunakan bahasa *Hardware Description Language* (HDL). FPGA dapat dikonfigurasikan untuk mengimplementasikan fungsi logika apapun yang dapat dilakukan oleh suatu *Application-specific IC* (ASIC). [8]

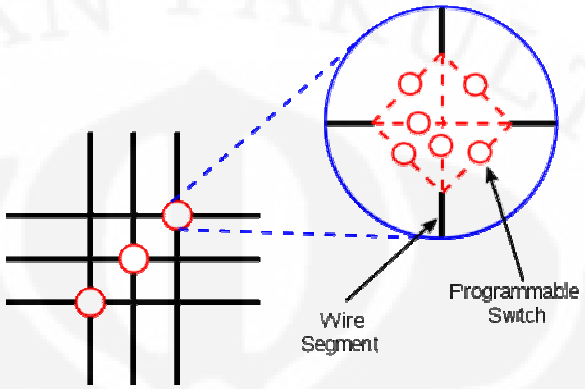
Karakteristik dari FPGA antara lain adalah dapat dirancang sesuai dengan keinginan dan kebutuhan pemakai tanpa melalui tahap "*burn*" di laboratorium atau di "*hardwire*" oleh perusahaan piranti, hal tersebut mungkin dilakukan karena FPGA terdiri atas sekumpulan *Configurable Logic Blocks* (CLB) yang terhubung melalui *Programmable Interconnects* (PI). Pada umumnya setiap CLB terdiri dari terdiri dari beberapa *Logic Cells* yang kadang disebut juga dengan *Slice*, yang masing-masing terdiri dari 4-input *Lookup Table* (LUT), D Flip-Flop dan 2-to-1 Mux, seperti pada gambar 2.1 dibawah ini.



Gambar 2.1 Sebuah Logic Cell pada FPGA [7]

Konfigurasi CLB dalam FPGA dapat berbeda-beda, tergantung dari manufaktur dan varian FPGA yang digunakan, perbedaannya termasuk jumlah *inputs* dan *outputs*, kompleksitas rangkaian CLB dan jumlah transistor yang digunakan. Jadi kemampuan untuk mengimplementasikan fungsi logika disediakan oleh CLB ini.

Setiap *Logic Cell* dapat dihubungkan dengan *Logic Cell* lainnya melalui *Programmable Interconnect* (PI) yang akan membentuk suatu fungsi logika yang compleks, ilustrasi dari PI ditunjukkan pada gambar 2.2 dibawah ini. [8]



Gambar 2.2 Programmable Interconnects [7]

2.2.2 Arsitektur FPGA

Sebuah IC FPGA terdiri 3 (tiga) komponen pendukung utamanya yakni [8]:

- *Configurable Logic Block* (CLB)

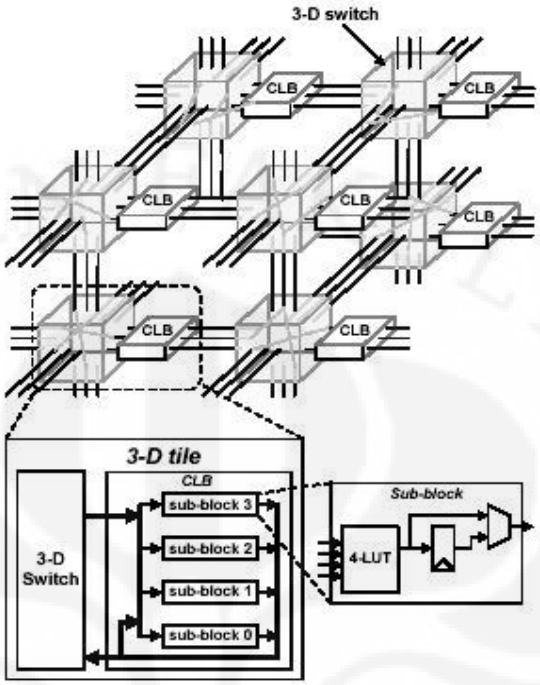
  CLB merupakan komponen dasar yang membentuk IC FPGA berupa matrik-matrik yang saling terhubung oleh PI, yang dapat dikonfigurasikan untuk melakukan rangkaian logika kombinasional, *shift register,* atau *Random Access Memory* (RAM).

- *Input Output Block* (IOB)

  IOB merupakan penghubung atau sebagai antarmuka antara pin-pin terminal IC FPGA dengan kawat penghubung atau jalur-jalur koneksi di luar IC, IOB dikelompokkan ke dalam beberapa *I/O Bank* yang sesuai dengan standar I/O.

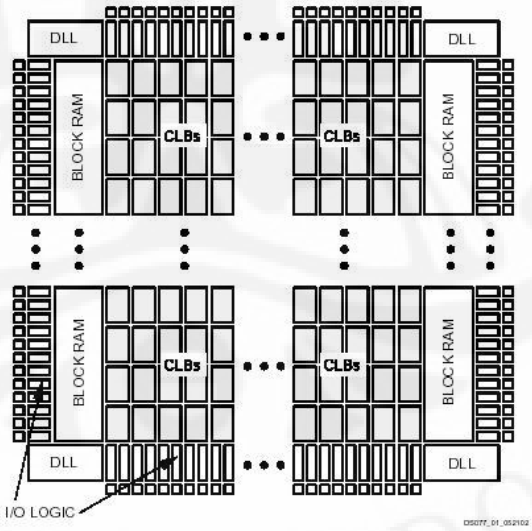
- *Programmable Interconnect* (PI).

  PI merupakan komponen yang berperan sebagai kawat atau sakelar penghubung yang dapat dikonfigurasikan dan mengelilingi blok-blok CLB, yang akan menghubungkan antar blok-blok CLB maupun dengan IOB.



Gambar 2.3 Arsitektur Dasar FPGA [15]

Selain CLB, IOB dan PI, beberapa IC FPGA juga dilengkapi dengan elemen-elemen lain seperti RAM dan Delay-Locked Loop (DLL) pada FPGA Spartan II series.



Gambar 2.4 Arsitektur FPGA Spartan II Series [14]

2.2.3 Mengkonfigurasikan FPGA

Tanpa memperhatikan metode interkoneksi yang digunakan pada FPGA, dapat dilihat jelas bahwa untuk membayangkan sakelar mana yang harus

dibuka dan yang ditutup untuk membuat suatu rangkaian logika merupakan pekerjaan yang sangat berat, karenanya perusahaan pembuat chip menyediakan perangkat lunak yang melakukan deskripsi dari disain logika sebagai masukannya dan kemudian akan menghasilkan file biner yang akan mengkonfigurasi sakelar-sakelar didalam chip CPLD atau FPGA sehingga menjadi seperti disain yang dibuat.

Perusahaan pembuat chip pada umumnya memberikan perangkat lunak secara cuma-cuma atau gratis. Perangkat lunak ini digunakan untuk mendukung proses *design entry*, *simulation, synthesis and place-and-route,* dan *Programming through special cables (JTAG).* Perusahaan Xilinx terkenal dengan software miliknya yang bernama ISE WebPack sedangkan perusahaan Altera terkenal dengan software miliknya yang bernama Quartus II Web Edition.

Implementasi sebuah disain logika dengan perangkat lunak tersebut biasanya terdiri dari beberapa langkah :

a. Mendeskripsikan rangkaian logika dari sistem yang dirancang dengan menggunakan *hardware description language* (HDL) seperti VHDL atau Verilog atau dengan menggambarkannya dengan *schematic editor.*
b. Deskripsi atau skematik rangkaian logika dari sistem dikonversikan menjadi sebuah *Netlist* menggunakan program *logic synthesizer. Netlist* merupakan sebuah deskripsi dari bermacam-macam gerbang logika dalam disain dan bagaimana mereka dihubungkan.
c. Implementasi rangkaian dengan melakukan pemetaan (*mapping*) gerbang logika dan interkoneksi (*routing*) netlist kedalam FPGA menggunakan *implementation tools*, yakni CLB didalam FPGA selanjutnya disusun kedalam *look-up tables (LUT)* yang melakukan operasi logika. CLB dan LUT terjalin dengan berbagai *routing resources*. Program pemetaan dan interkoneksi menggumpulkan netlis gerbang logika kedalam kelompok yang muat dalam LUT kemudian menentukan kumpulan gate ke CLB tertentu sambil membuka atau

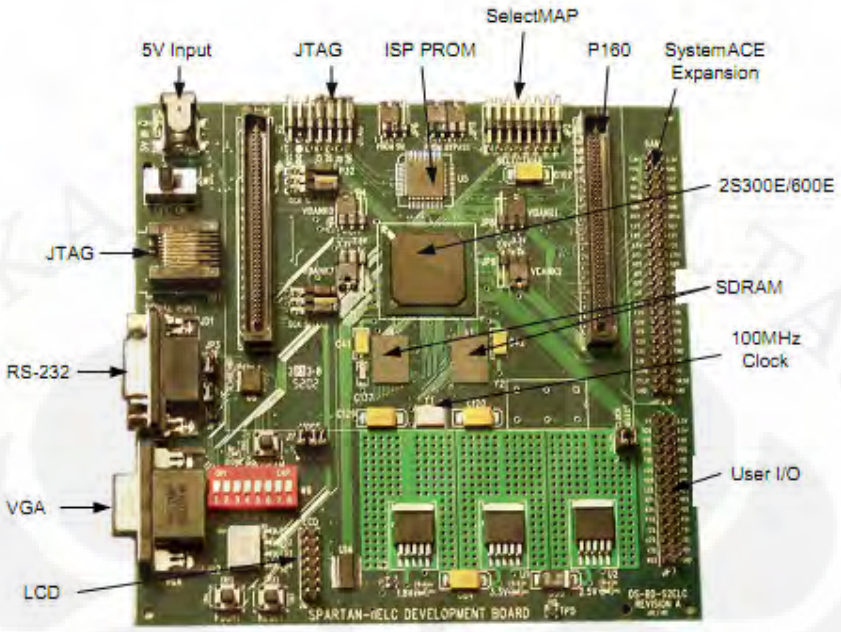
menutup sakelar pada matriks interkoneksi untuk menghubungkan gerbang-gerbang logika tersebut

d. Setelah fase implementasi selesai, aplikasi akan membuat sebuah *file bitstream* yang berisi nilai logika '1' dan '0' yang sebagai representasi konfigurasi rangkaian digital yang akan diimplementasikan ke dalam FPGA untuk menunjukkan terbuka atau tertutupnya sakelar pada IC FPGA.
e. *File bitstream* kemudian digugah (*upload*) kedalam IC FPGA atau IC memori yang tersedia melalui kabel JTAG sesuai dengan jenis dan tipe *development board* yang digunakan.

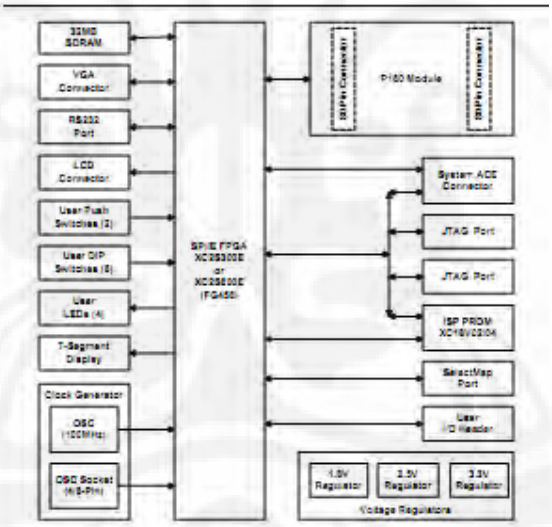
2.2.4 Spesifikasi FPGA Xilinx Spartan-IIE LC Development Board.

Spesifikasi FPGA Xilinx Spartan-IIE LC Development Board yang digunakan pada tugas akhir ini adalah sebagai berikut [9]:

a. Memiliki FPGA Xilinx Spartan-IIE XC2S300E-6FG456C dengan 300.000 gerbang logika, yang terdiri dari 3072 buah CLB, 6144 buah Flip-Flop, 6144 buah 4-input LUT dan 329 *I/O Block* .
b. Internal Clock source : 100 MHz
c. SDRAM 32 Mb.
d. Antarmuka simple "R-2R" *resistor-ladder* VGA
e. Konektor DB9 RS232 sederhana hanya untuk pin Td dan Rd dengan IC MAX3221 dari Texas Instruments.
f. konektor LCD 8-bit 2x16 karakter
g. 4 buah LED dan 1 (satu) buah seven segment display
h. 8 bit user DIP switch.
i. 82 pin *header* yang dapat digunakan oleh pengguna melalui 2 (dua) buah *on-board* header.
j. P160 *Expansion Module Standard*
k. System ACE *Expansion* untuk aplikasi *Real Time Operating System* (RTOS) dari eksternal *CompactFlash*
l. Antarmuka RJ45 untuk menggugah *file bitstream* konfigurasi.
m. XC18V02 ISP PROM untuk menyimpan konfigurasi.



Gambar 2.5 Xilinx FPGA Spartan-IIE LC Development Board [9]



Gambar 2.6 Blok Diagram Spartan-IIE LC Development Board [9]

2.3 VHSIC Hardware Description Languange (VHDL)

2.3.1 Pengertian

VHDL merupakan salah satu bahasa yang digunakan untuk mendeskripsikan suatu perangkat keras. VHDL merupakan kependekan dari VHSIC *Hardware Description Languange*, sedangkan VHSIC kependekan dari *Very High Speed Integrated Circuit*.

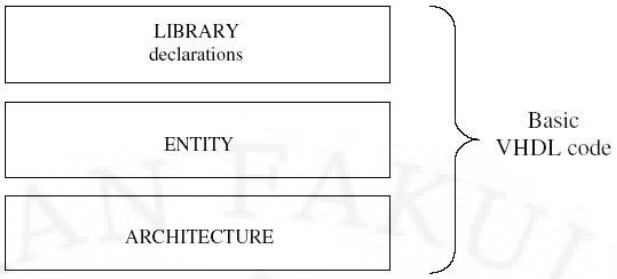
Pada awalnya VHDL merupakan standard dokumentasi perangkat keras yang disponsori oleh Departemen Pertahanan Amerika Serikat sekitar tahun 1980, yang kemudian dokumentasi tersebut dikirimkan ke *Institute of Electrical and Electronic Engineers* (IEEE) dan diratifikasi oleh IEEE dengan standard IEEE 1076 dan IEEE 1164 dengan nama VHDL-87. [13]

Setelah peluncuran pertama, berbagai varian VHDL dikembangkan untuk memfasilitasi berbagai macam kebutuhan perancangan dan pemodelan perangkat keras, varian-varian tersebut juga mendapatkan standard dari IEEE, yakni sebagai berikut :

- Standard IEEE 1076.1-1999, VHDL Analog and Mixed Signal Exensions (VHDL AMS), menggambarkan pemodelan sinyal analog dan campuran.
- Standard IEEE 1076.2-1996, VHDL Mathematical Packages, menggambarkan fungsi matematika untuk bilangan real dan kompleks.
- Standard IEEE 1076.3-1997, Synthesis Packages, menggambarkan fungsi aritmatika dengan manipulasi bit.
- Standard IEEE 1076.4-1995, VHDL Initiative Towards ASIC Lybraries (VITAL), menggambarkan mekanisme penambahan informasi detil pewaktuan sel-sel pada ASIC.
- Standard IEEE 1076.6-1999, VHDL Register Transfer Level (RTL) Synthesis, menggambarkan VHDL yang kompatibel dengan proses Synthesis.
- Standard IEEE 1164-1993, Multivalue Logic System For VHDL Model Interoperability (std_logic_1164), menggambarkan tipe data yang baru untuk memodelkan logika dengan beragam nilai.
- Standard IEEE 1029.1-1998, VHDL Waveform and Vector Exchange to Support Design and Test Verification (WAVES), menggambarkan penggunaan VHDL dalam pertukaran informasi pada lingkungan simulasi HDL.

2.3.2 Struktur Dasar

Secara umum, struktur bahasa pemograman VHDL terdiri dari tiga bagian pokok [1], yaitu deklarasi *Library*, *Entity* dan *Architecture*.



Gambar 2.7 Struktur Dasar VHDL [1]

Deklarasi *Library* merupakan kumpulan potongan kode yang sering digunakan, Sebuah *entity* berisi deklarasi *Input/Output* dari suatu system, sedangkan deklarasi *architecture* berisi deskripsi dari suatu rangkaian atau fungsi logika yang akan diimplemntasikan. Di dalam *architecture* dapat menangani baik rangkaian sekuensial maupun rangkaian kombinasional dan dalam 1 (satu) modul VHDL dimungkinkan untuk menggunakan lebih dari 1 (satu) *architecture.*

2.3.3 Metode Verifikasi

Metode verifikasi yang sering digunakan untuk mengetahui hasil implementasi rancangan dengan VHDL adalah dengan melakukan simulasi. Simulasi merupakan proses rekonstruksi sistem yang telah dideskripsikan dengan VHDL kemudian mengeksekusi sistem tersebut dengan data masukan dengan pola tertentu pada perangkat komputer yang kemudian menguji dan menganalisa respon dari sistem melalui keluarannya. Keluaran yang diuji dan dianalisa bisa berdasarkan sistem yang sesungguhnya (sesuai dengan datasheet) atau berdasarkan nilai ekspektasi (sesuai dengan teori atau hipotesis dari keluarannya) yang menyertakan informasi fungsional dan pewaktuannya.

Simulasi merupakan proses yang serbaguna karena dapat diaplikasikan pada level abstraksi apapun baik pada level skematik amupun kode VHDL. Manfaat utama dari simulasi adalah memberikan fasilitas kepada pengguna untuk menguji dan mendeteksi kesalahan-kesalahan pada rancangan tanpa benar-benar merekonstruksi rancangan. Tetapi simulasi hanya memberikan sebagian potongan informasi dari keseluruhan operasi pada rancangan, yang digambarkan oleh serangkaian input stimulus yang

tidak dijamin mencakup seluruh kemungkinan stimulus yang dibutuhkan oleh sistem, sehingga simulasi hanya mampu mendeteksi kesalahan-kesalahan utama saja.

Keterbatasan simulasi lainnya adalah kompleksitas komputasi yang dibutuhkan, perangkat keras dapat bekerja secara kombinasional dan parallel sehingga membutuhkan cukup banyak waktu jika dioperasikan pada komputer yang melakukan komputasi secara sekuensial, yang akan menjadi masalah serius ketika harus melakukan simulasi dengan sistem yang terdiri dari ratusan hingga ribuan komponen level rendah.

Metode verifikasi lainnya selain simulasi adalah analisa waktu, verifikasi formal dan emulasi perangkat keras [3]. Analisa waktu hanya berfokus pada perhitungan waktu yang dibutuhkan pada setiap proses dan semua kemungkinan delay agar waktu yang dibutuhkan oleh sistem sesuai dengan spesifikasi waktu yang dibutuhkan.

Verifikasi formal adalah memverifikasi sistem berdasarkan kesesuaian hasil operasi pada sistem dari sisi perhitungan matematika, apakah nilai keluarannya sudah sesuai dengan rancangan yang diimplementasikan.

Emulasi perangkat keras adalah melakukan rekonstruksi sistem ke dalam FPGA atau menggugah konfigurasi yang dihasilkan oleh VHDL ke dalam FPGA dan memverifikasi fungsional FPGA seperti halnya memverifikasi FPGA sebagai ASIC.

2.3.4 Metode Deskripsi VHDL

Deskripsi dengan VHDL dapat dilakukan dengan beberapa metode, baik digunakan dengan salah satu metode atau campuran dari beberapa metode. Metode deskripsi VHDL antara lain metode struktural, aliran data dan perilaku sistem. Metode struktural mendeskripsikan sistem sebagai sebuah struktur yang terbentuk dari komponen-komponen yang dibutuhkan dan bagaimana masing-masing komponen tersebut saling terhubung, atau kurang lebih seperti membuat skematik dari sistem.

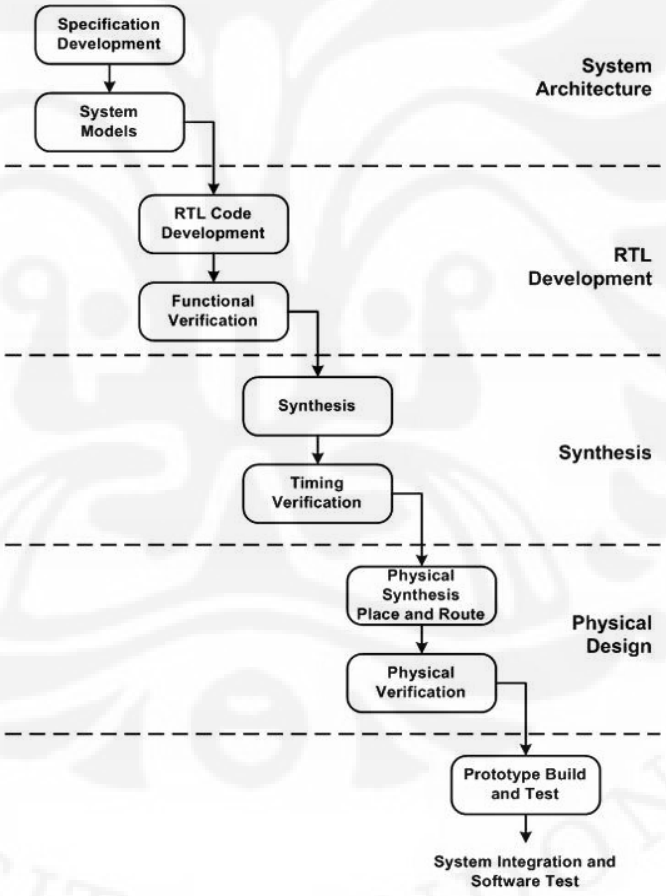
Metode aliran data menitikberatkan pada bagaimana data itu mengalir dari satu komponen ke komponen lain, dan bagaimana data tersebut berubah setelah melewati komponen demi komponen. Komponen-komponen yang

dilewati baik rangkaian logika maupun rangkaian memori atau register, sehingga lebih sering disebut sebagai *Register Transfer Level.*

Sedangkan metode perilaku sistem mendeskripsikan sistem hanya dari sisi perilaku sistem atau kesesuaian antara input dan output yang dihasilkan, metode ini tidak memperhatikan komponen atau rangkaian logika serta memori yang digunakan, sehingga jika digunakan untuk rancangan dengan kompleksitas tinggi dapat menyebabkan pemborosan atau ketidaksesuaian penggunaan *source* yang tersedia.

2.4 Metode Perancangan

2.4.1 Waterfall ASIC Design Flow



Gambar 2.8 Waterfall ASIC Design Flow [9]

Waterfall ASIC design flow adalah bagian dari *Reuse Methodology Manual (RMM)* [9] yang merupakan suatu metodologi perancangan sistem

on-chip atau ASIC, waterfall ASIC design flow merupakan metode yang paling sederhana. Dalam waterfall ASIC design flow produk tiap-tiap tahapannya harus selesai atau dilengkapi dengan sempurna karena tidak jalan untuk kembali ke tahap sebelumnya. Tahap-tahap perancangannya ditunjukkan oleh gambar 2.8

2.4.1.1 System Architecture

Pada tahap ini perancang harus mampu menganalisa dan menguraikan spesifikasi sistem secara tepat dan detil termasuk mengidentifikasi kebutuhan-kebutuhan dari sistem. Selanjutnya spesifikasi tersebut harus dapat divisualisasikan dengan teknik pemodelan tertentu sehingga rancangan tersebut bisa direpresentasikan dengan sebuah arsitektur dari sistem tersebut.

Arsitektur yang dibuat harus mencakup semua komponen yang dibutuhkan sistem, termasuk interkoneksi dengan blok-blok sistem yang lain dan jumlah aliran data atau pin untuk masukan dan keluaran.

2.4.1.2 RTL Development

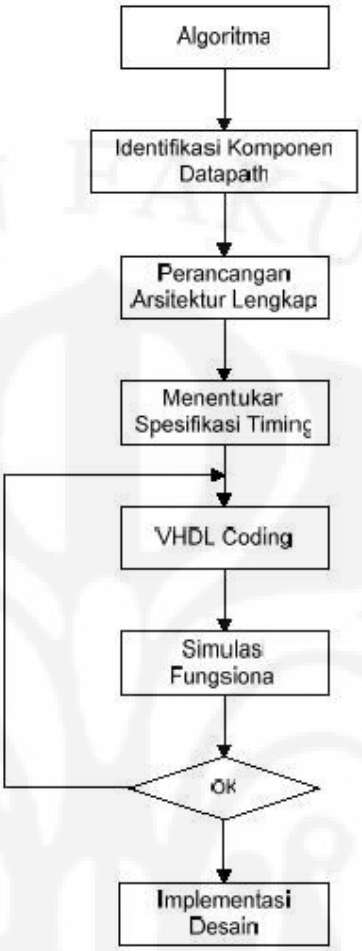
Produk yang diharapkan dari tahapan ini adalah serangkaian persamaan boolean/logika yang sudah terintegrasi dan telah berhasil melewati simulasi fungsional. Sebelum melakukan penulisan kode tersebut, perancang harus menganalisa dan menguraikan fungsi dan proses yang harus dilakukan oleh tiap-tiap blok dari sistem tersebut atau dengan kata lain mendapatkan persamaan boolean/logika yang paling optimal, termasuk melakukan simulasi fungsional baik dengan perangkat simulasi atau dengan membuat kode testbench.

2.4.1.3 Synthesis, Physical Design dan System integration

Ketiga tahap ini dilakukan oleh perangkat lunak implementasi tergantung dengan jenis dan tipe programmable logic device yang digunakan.

2.4.2 Perancangan Sistem dengan VHDL

Perancangan sistem dengan VHDL menggunakan pendekatan secara *top down*, tahap-tahap perancangannya ditunjukkan dengan diagram alir berikut [4];



Gambar 2.9 Diagram Alir Perancangan Sistem dengan VHDL [4]

2.4.2.1 Algoritma

Tahap pertama yang dilakukan adalah melakukan analisa dan menentukan spesifikasi sistem tersebut, dalam tahap ini diharapkan perancang mampu mendeskripsikan dan menguraikan komponen-komponen yang dibutuhkan oleh sistem. Secara garis besar ada 4 (empat) katagori komponen yang harus diidentifikasi, yakni

- Simpan, yakni media penyimpanan atau memori yang dibutuhkan oleh sistem, termasuk jenis, tipe dan besarnya data yang akan disimpan serta jenis media penyimpan yang dibutuhkan, misalnya ROM untuk data yang statis atau RAM untuk data yang dinamis.
- Proses, komponen proses apa saja yang dibutuhkan. Komponen yang memproses masukan sedemikian rupa hingga menghasilkan keluaran.

Komponen proses tersebut juga harus diperhatikan perilakunya, apakah komponen proses termasuk kombinasional atau sekuensial.

- Kendali, selain menyimpan dan memproses data suatu sistem juga memerlukan komponen pengendali (controller) untuk mengatur kerja dari masing-masing blok sistem atau komponen lain. Komponen ini sangat penting karena akan menentukan jenis dan lebar jalur/bus yang digunakan.
- Transfer, perancang harus memperhatikan komponen yang bertugas untuk transfer/perpindahan data, terutama perpindahan data dari dan ke luar sistem. Komponen ini merupakan modul antarmuka sistem yang akan berhubungan dengan perangkat lain di luar sistem.

2.4.2.2 Identifikasi Komponen

Melakukan identifikasi terhadap komponen-komponen yang telah diuraikan pada tahap sebelumnya, komponen apa saja yang merupakan komponen yang sudah standar (seperti full adder , decoder, multiplexer dan lainnya) atau telah tersedia dalam library (IP cores) FPGA yang digunakan. Pada tahap ini tiap-tiap komponen harus sudah merepresentasikan suatu komponen yang utuh dan lengkap dengan pin-pin masukan dan keluaran. Untuk mempermudah identifikasi komponen biasanya pada tahap ini dilakukan pemodelan terhadap rancangan.

2.4.2.3 Perancangan Arsitektur

Setelah tiap-tiap komponen telah teridentifikasi secara utuh, maka tahap berikutnya adalah mengintegrasikan seluruh komponennya sesuai dengan rancangan sistem termasuk sistem jalur/bus yang digunakan, sehingga terbentuk sebuah arsitektur lengkap dan detil dari rancangan tersebut.

2.4.2.4 Spesifikasi Waktu

Pada tahap ini setiap komponen harus mampu dikalkulasi waktu yang dibutuhkan untuk menyelesaikan fungsinya, terutama komponen-komponen proses yang sekuensial. Akan lebih sempurna jika juga memperhitungkan delay. Pada akhirnya rancangan tidak hanya benar-benar terintegrasi dalam arsitekturnya tetapi juga data yang mengalir dalam arsitektur tersebut telah sinkron satu dengan lainnya.

2.4.2.5 Rekonstruksi Kode VHDL

Dalam rekonstruksi kode VHDL banyak memberikan pilihan teknik deskripsi. Untuk komponen-komponen yang standar atau telah tersedia pada aplikasi dari vendor maka rekonstruksi bisa dilakukan secara blok skematik atau struktural, untuk komponen rangkaian kombinasional sederhana bisa menggunakan teknik RTL dengan bantuan Karnough Map, sementara untuk rangkaian sekuensial sederhana bisa menggunakan teknik RTL dengan bantuan Finite State Machine. Sedangkan untuk komponen yang lebih rumit atau kompleksitas tinggi bisa menggunakan pendekatan teknik perilaku sistem

2.4.2.6 Simulasi Fungsional

Simulasi fungsional akan sangat membantu jika setiap komponen disimulasikan satu persatu untuk mempermudah pelacakan kesalahan bila terjadi masalah dalam sistem. Pembuatan kode testbench untuk masing komponen juga akan sangat membantu, agar test vector yang digunakan untuk simulasi lebih konsisten.

2.4.2.7 Implementasi Disain

Implementasi disain dilakukan menggunakan perangkat lunak sesuai dengan jenis dan tipe FPGA yang digunakan

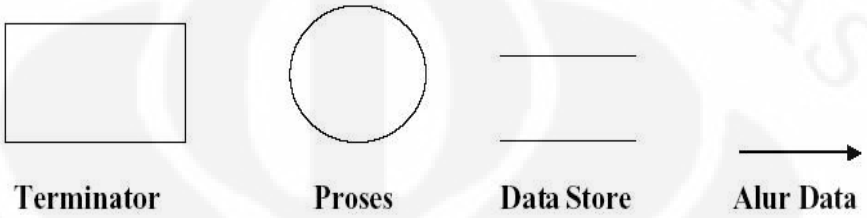
2.4.3 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama Bubble chart, Bubble diagram, model proses, diagram alur kerja, atau model fungsi. DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program [11].

2.4.3.1 Komponen DFD

Komponen DFD terdiri 4 (empat) komponen dengan notasi seperti yang ditunjukkan oleh gambar 2.10



Gambar 2.10 Komponen DFD [11]

Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Biasanya terminator dikenal dengan nama entitas luar (*external entity*). Terminator dapat berupa orang, sekelompok orang, organisasi, departemen di dalam organisasi, atau perusahaan yang sama tetapi di luar kendali sistem yang sedang dibuat modelnya. Terminator dapat juga berupa departemen, divisi atau sistem di luar sistem yang berkomunikasi dengan sistem yang sedang dikembangkan. Komponen terminator ini perlu diberi nama sesuai dengan dunia luar yang berkomunikasi dengan sistem yang sedang dibuat modelnya, dan biasanya menggunakan kata benda, misalnya *Bagian Penjualan, Dosen, Mahasiswa*.

Ada tiga hal penting yang harus diingat tentang terminator :

- Terminator merupakan bagian/lingkungan luar sistem. Alur data yang menghubungkan terminator dengan berbagai proses sistem, menunjukkan hubungan sistem dengan dunia luar.
- Profesional Sistem Tidak berhak mengubah isi atau cara kerja organisasi atau prosedur yang berkaitan dengan terminator
- Hubungan yang ada antar terminator yang satu dengan yang lain tidak digambarkan pada DFD.

Komponen *Proses* menggambarkan bagian dari sistem yang mentransformasikan input menjadi output. Proses diberi nama untuk menjelaskan proses/kegiatan apa yang sedang/akan dilaksanakan. Pemberian nama proses dilakukan dengan menggunakan kata kerja transitif (kata kerja yang membutuhkan obyek), seperti *Menghitung Gaji, Mencetak KRS, Menghitung Jumlah SKS*.

Ada beberapa hal yang perlu diperhatikan tentang proses :

- Proses harus memiliki input dan output.
- Proses dapat dihubungkan dengan komponen terminator, data store atau proses melalui alur data.
- Sistem/bagian/divisi/departemen yang sedang dianalisis oleh profesional sistem digambarkan dengan komponen proses.

Komponen *Data Store* digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak, misalnya *Mahasiswa*. Data store ini biasanya berkaitan dengan penyimpanan-penyimpanan, seperti file atau database yang berkaitan dengan penyimpanan secara komputerisasi, misalnya file disket, file harddisk, file pita magnetik. Data store juga berkaitan dengan penyimpanan secara manual seperti buku alamat, file folder, dan agenda.

Alur Data digambarkan dengan anak panah, yang menunjukkan arah menuju ke dan keluar dari suatu proses. Alur data ini digunakan untuk menerangkan perpindahan data atau paket data/informasi dari satu bagian sistem ke bagian lainnya. Selain menunjukkan arah, alur data pada model yang dibuat oleh profesional sistem dapat merepresentasikan bit, karakter, pesan, formulir, bilangan real, dan macam-macam informasi yang berkaitan dengan komputer. Alur data juga dapat merepresentasikan data/informasi yang tidak berkaitan dengan komputer. Alur data perlu diberi nama sesuai dengan data/informasi yang dimaksud, biasanya pemberian nama pada alur data dilakukan dengan menggunakan kata benda, contohnya *Laporan Penjualan*.

2.4.3.2 Penggambaran DFD

Tidak ada aturan baku untuk menggambarkan DFD, tapi dari berbagai referensi yang ada, secara garis besar langkah untuk membuat DFD adalah:

- Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem.
- Identifikasi semua input dan output yang terlibat dengan entitas luar.
- Buat Diagram Konteks (diagram context), diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya. Dalam menggambarkan diagram konteks sebelumnya harus menentukan nama dan batasan sistem, terminator apa saja yang ada dalam sistem, serta data yang mengalir antara sistem dan terminator.
- Buat Diagram Level Zero, diagram ini adalah dekomposisi dari diagram konteks. Sebelum menggambarkannya terlebih dahulu tentukan proses utama yang ada pada sistem dan masukan/keluaran dari masing-masing proses dengan memperhatikan konsep keseimbangan (alur data yang keluar/masuk dari suatu level harus sama dengan alur data yang masuk/keluar pada level berikutnya). Apabila diperlukan, munculkan data store (master) sebagai sumber maupun tujuan alur data. Pada semua level (dari level zero sampai level terakhir) sebaiknya hindari perpotongan arus data dan beri nomor pada proses utama (nomor tidak menunjukkan urutan proses).
- Buat Diagram Level Satu, diagram level satu merupakan dekomposisi dari diagram level zero dengan menentukan proses yang lebih kecil (sub-proses) dari proses utama yang ada di level zero kemudian tentukan apa yang diberikan/diterima masing-masing sub-proses ke/dari sistem dan perhatikan konsep keseimbangan. Apabila diperlukan, munculkan data store (transaksi) sebagai sumber maupun tujuan alur data. Beri nomor pada masing-masing sub-proses yang menunjukkan dekomposisi dari proses sebelumnya. Contoh : 1.1, 1.2 dekomposisi dari proses 1, 2.1, 2.2 dekomposisi dari proses 2 dan seterusnya.

- DFD Level Dua, Tiga, dan seterusnya merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level satu.

2.4.3.3 Syarat-Syarat Pembuatan DFD

Syarat pembuatan DFD ini akan menolong profesional sistem untuk menghindari pembentukkan DFD yang salah atau DFD yang tidak lengkap atau tidak konsisten secara logika. Beberapa syarat pembutan DFD dapat menolong profesional sistem untuk membentuk DFD yang benar, menyenangkan untuk dilihat dan mudah dibaca oleh pemakai.

Syarat-syarat pembuatan DFD ini adalah :

- Pemberian nama untuk tiap komponen DFD

  Komponen-komponen tersebut perlu diberi nama yang tepat, agar siapa yang membaca DFD khususnya pemakai akan merasa yakin bahwa DFD yang dibentuk ini adalah model yang akurat.
- Pemberian nomor pada komponen proses

  Biasanya profesional sistem memberikan nomor dengan bilangan terurut pada komponen proses sebagai referensi. Tidak jadi masalah bagaimana nomor-nomor proses ini diberikan. Nomor proses dapat diberikan dari kiri ke kanan, atau dari atas ke bawah, atau dapat pula dilakukan dengan pola-pola tertentu selama pemberian nomor ini tetap konsisten pada nomor yang dipergunakan.
- Penggambaran DFD sesering mungkin agar enak dilihat

  Penggambaran DFD dapat dilakukan berkali-kali sampai secara teknik DFD itu benar, dapat diterima oleh pemakai, dan sudah cukup rapih sehingga profesional sistem tidak merasa malu untuk menunjukkan DFD tersebut. Dengan kata lain, penggambaran DFD ini dilakukan sampai terbentuk DFD yang enak dilihat, dan mudah dibaca oleh pemakai dan profesional sistem lainnya.
- Penghindaran penggambaran DFD yang rumit

  Tujuannya adalah agar model yang dibuat itu mudah dibaca dan dimengerti tidak hanya oleh profesional sistem yang membuat DFD,

tetapi juga oleh pemakai yang berpengalaman dengan subyek yang terjadi. Hal ini berarti DFD harus mudah dimengerti, dibaca, dan menyenangkan untuk dilihat.

- Penggambaran DFD yang dibentuk itu konsiten secara logika
  Penggambaran DFD harus konsisten terhadap kelompok DFD lainnya. Profesional sistem menggambarkan DFD berdasarkan tingkatan DFD dengan tujuan agar DFD yang dibuatnya itu mudah dibaca dan dimengerti oleh pemakai sistem. Hal ini sesuai dengan salah satu tujuan atau syarat membuat DFD.

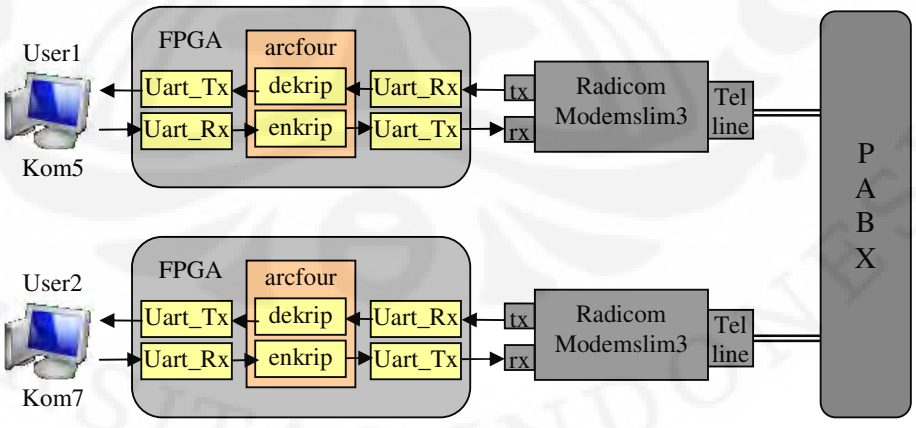
BAB 3
PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Deskripsi dan Spesifikasi Sistem

3.1.1 Deskripsi Sistem

Rancangan sistem pengamanan data pada tugas akhir ini merupakan implementasi algoritma enkripsi arcfour ke dalam FPGA Xilinx Spartan-IIE LC Development Board sebagai modul kripto untuk mengamankan perpindahan data khususnya data berupa teks dan file melalui perangkat PABX dan Radicom Modemslim3, sedangkan antarmuka pengguna untuk memasukkan dan menampilkan data teks melalui komputer personal menggunakan aplikasi hyperterminal private edition.

Data yang akan ditransmisikan baik berupa teks atau file akan dimasukkan melalui hyperterminal yang kemudian dienkripsi oleh FPGA Xilinx Spartan-IIE LC Development Board kemudian ditransmisikan menggunakan modem melalui PABX sebagai representasi dari PSTN, di sisi penerima data yang telah dienkripsi diterima oleh modem dan akan didekripsi oleh FPGA sehingga data yang dikirimkan kembali ke bentuk semula. Skema rancangan ditunjukkan pada gambar 3.1 dibawah ini.



Gambar 3.1 Skema Rancangan

Komputer personal akan dihubungkan dengan modul kripto menggunakan kabel *USB to Serial,* dengan tegangan pada level TTL karena baik FPGA Xilinx Spartan-IIE LC Development Board maupun Radicom Modemslim3 beroperasi pada tegangan level TTL.

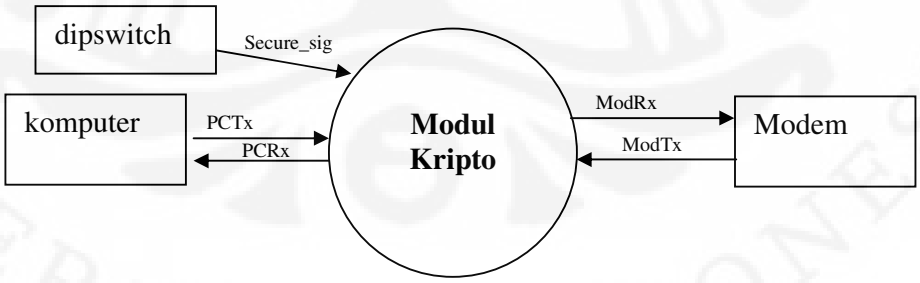
3.1.2 Spesifikasi Sistem

Komponen-komponen yang dibutuhkan oleh modul kripto terdiri dari 3 (tiga) bagian utama yakni UART_Rx, UART_Tx dan Arcfour Cryptoprocessor . Komponen UART_Rx merupakan antarmuka antara modul kripto dengan komputer personal dan modem untuk dapat menerima data yang dikirimkan oleh kedua perangkat tersebut, sedangkan UART_Tx digunakan sebagai antarmuka dengan komputer personal dan modem untuk dapat mengirimkan data ke kedua perangkat tersebut. Arcfour Cryptoprocessor berperan untuk melakukan proses enkripsi dan dekripsi data yang melalui modul kripto.

komponen UART_Rx dan UART_Tx menggunakan baudrate 57.600 bps , 1 bit startbit, 8 bit data, 1 bit stopbit tanpa parity dan *flow control* untuk melakukan komunikasi dengan personal komputer maupun dengan modem.

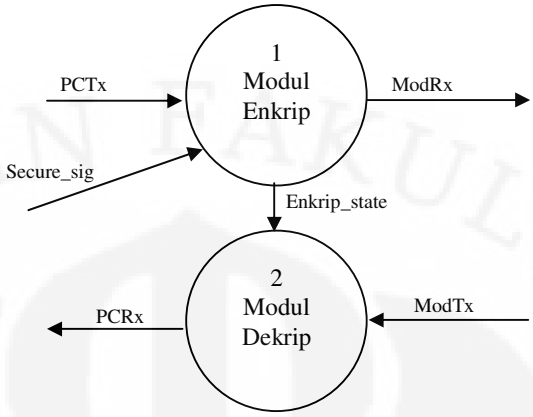
3.2 Pemodelan Sistem

3.2.1 Diagram Konteks



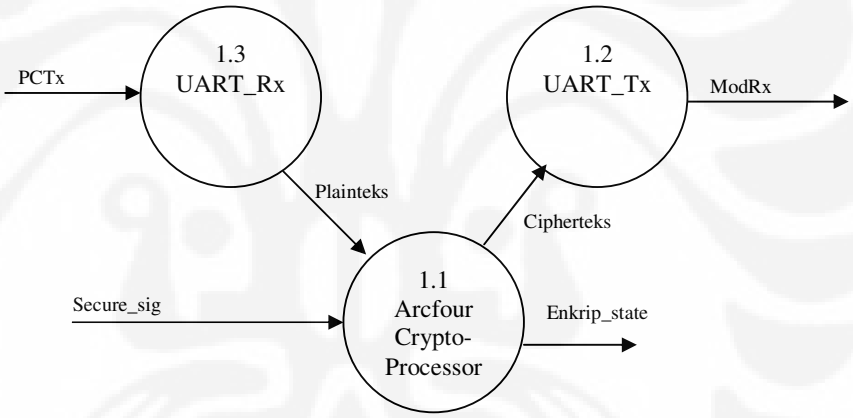
Gambar 3.2 Conteks Diagram Modul Kripto

3.2.2 DFD Level Nol



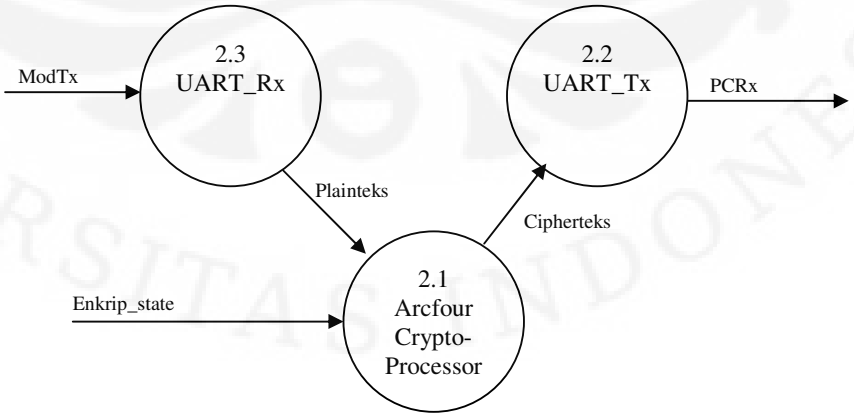
Gambar 3.3 DFD Level Nol

3.2.3 DFD Level Satu Modul Enkrip



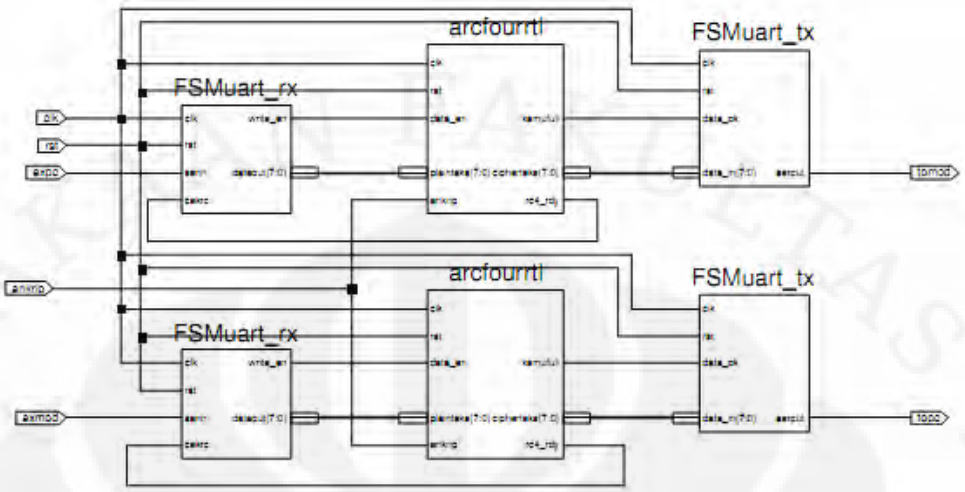
Gambar 3.4 DFD Level Satu Modul Enkrip

3.2.4 DFD Level Satu Modul Dekrip



Gambar 3.5 DFD Level Satu Modul Dekrip

3.3 Perancangan Arsitektur Sistem



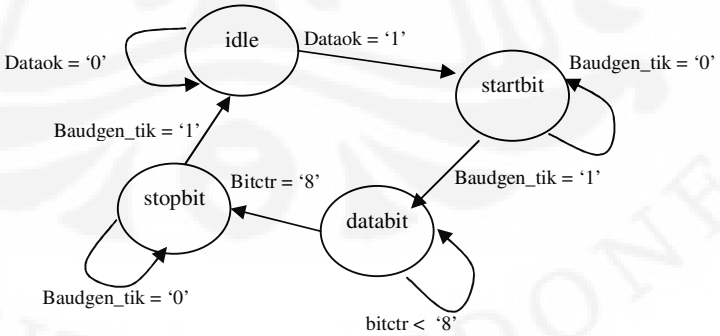
Gambar 3.6 Arsitektur Modul Enkripsi

3.4 Perancangan Sistem Dengan VHDL

3.4.1 Perancangan modul VHDL UART_Tx

Modul VHDL UART_Tx berfungsi untuk melakukan penguraian data parallel 8 bit dari modul Arcfour Cryptoprocessor ke komputer personal atau modem, untuk itu dibutuhkan komponen-komponen seperti parallel to serial converter, baudrate generator dan sebuah blok kendali.

Untuk merekonstruksi Parallel to serial converter menggunakan shift register, untuk baudrate generator menggunakan metode sampling, untuk blok kendali dapat ditunjukkan dengan state diagram dibawah ini.



Gambar 3.7 UART_Tx State Diagram

Pada state idle modul UART_Tx menunggu sinyal dataok dari modul Arcfour Cryptoprocessor bernilai '1' yang memberi informasi bahwa data parallel 8 bit telah siap diurai, pada state startbit modul mengirimkan bit '0' sebagai startbit

sampai sinyal baudgen_tik bernilai ‘1’, kemudian pada state databit modul menguraikan data parallel 8 bit menjadi serial dengan mengirimkan LSB terlebih dahulu dan berhenti saat bitctr bernilai ‘8’. Pada state stopbit modul mengirimkan bit ‘1’ sebagai stopbit.

Kode VHDL untuk modul VHDL UART_Tx :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FSMuart_tx is
   Port ( rst : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          data_ok : in  STD_LOGIC;
          data_in : in  STD_LOGIC_VECTOR (7 downto 0);
          serout : out  STD_LOGIC);
end FSMuart_tx;

architecture Behavioral of FSMuart_tx is
signal tik : std_logic;
type txstate is (idle, startbit, databit, stopbit);
signal akustate : txstate;
--================================================
--1736,hyperterminal 57600 bps, FPGA 100 MHz
--================================================
signal baud96tx : std_logic_vector(10 downto 0);
constant baudclk : std_logic_vector(10 downto 0) := "11011001000";
signal bitctr: std_logic_vector(2 downto 0);
signal parseout : std_logic_vector(7 downto 0);
begin
--================================================
-- Baudrate generator
--================================================
baudgen: process(rst,clk,akustate,baud96tx)
```

```
begin
if rst = '1' then
        baud96tx <= (others => '0');
        tik <= '0';
elsif rising_edge(clk) then
        if akustate = idle then
                tik <= '0';
                baud96tx <= (others => '0');
        else
                tik <= '0';
                baud96tx <= baud96tx + 1;
                if baud96tx = baudclk then
                        baud96tx <= (others => '0');
                        tik <= '1';
                end if;
        end if;
end if;
end process;
--=============================================
-- State Machine UART_Tx
--=============================================
txFSM: process(rst,clk,tik,data_ok,bitctr)
begin
if rst = '1' then
        akustate <= idle;
        parseout <= (others => '1');
        bitctr <= "000";
        serout <= '1';
elsif rising_edge(clk) then
        case akustate is
        when idle =>
        serout <= '1';
```

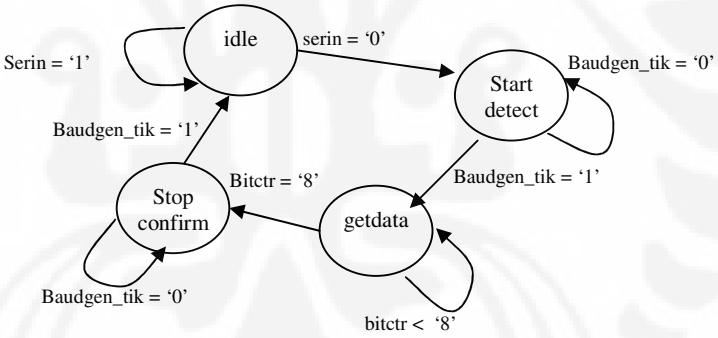
```
bitctr <= "000";
if data_ok = '1' then
        akustate <= startbit;
        parseout <= data_in;   -- xor kuncirc4;
else
        akustate <= idle;
end if;
when startbit =>
serout <= '0';
if tik = '1' then
        akustate <= databit;
else
        akustate <= startbit;
end if;
        when databit =>
serout <= parseout(0);
if tik = '1' then
        parseout <= '1' & parseout(7 downto 1);
        if bitctr = "111" then
                akustate <= stopbit;
        else
                bitctr <= bitctr + 1;
                akustate <= databit;
        end if;
else
        akustate <= databit;
        parseout <= parseout;
        bitctr <= bitctr;
end if;
when stopbit =>
serout <= '1';
akustate <= idle;
```

```
        when others =>
        akustate <= idle;
        end case;
end if;
end process;
end Behavioral;
```

3.4.2 Perancangan modul VHDL UART_Rx

Modul VHDL UART_Rx berfungsi untuk melakukan akuisisi data serial dari komputer personal atau modem dan mengirimkan data parallel 8 bit ke modul arcfour cryptoprocessor, untuk itu dibutuhkan komponen-komponen seperti serial to parallel, baudrate generator dan sebuah blok kendali.

Untuk merekonstruksi serial to parallel converter menggunakan shift register, untuk baudrate generator menggunakan metode sampling, untuk blok kendali dapat ditunjukkan dengan state diagram dibawah ini.



Gambar 3.8 UART_Rx State Diagram

Pada saat state idle modul menunggu startbit dari serin, kemudian pada start detect modul mengakuisisi nilai start bit sekaligus memverifikasi bahwa start bit telah teridentifikasi sebagai bit '0'. State getdata melakukan pengumpulan data serial dari komputer personal atau modem untuk dijadikan data parallel 8 bit, saat state stop confirm modul mengkonfirmasi data telah selesai dengan adanya stopbit sebagai bit '1' sekaligus mengirimkan data parallel 8 bit kepada modul arcfour cryptoprocessor.

Kode VHDL untuk modul UART_Rx :

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity FSMuart_rx is
    Port ( clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           serin : in  STD_LOGIC;
           write_en : out  STD_LOGIC;
           dataout : out  STD_LOGIC_VECTOR (7 downto 0));
end FSMuart_rx;

architecture Behavioral of FSMuart_rx is
signal tik : std_logic;
type rxstate is (idle, startdetect, getdata, stopconfirm);
signal akustate : rxstate;
--================================================
-- 1736, hyperterminal 57600 bps, FPGA 100 MHz
--================================================
signal baud96rx : std_logic_vector(10 downto 0);
constant baudclk : std_logic_vector(10 downto 0) := "11011001000";
constant getbit : std_logic_vector(10 downto 0) := "01101100100"; --868
signal bitctr: std_logic_vector(2 downto 0);
signal modemout : std_logic_vector(7 downto 0);
begin
--================================================
-- Baudrate generator
--================================================
baudgen: process(rst,clk,akustate)
begin
if rst = '1' then
        baud96rx <= (others => '0');
        tik <= '0';
```

```
elsif rising_edge(clk) then
        if akustate = idle then
                tik <= '0';
                baud96rx <= (others => '0');
        else
                tik <= '0';
                baud96rx <= baud96rx + 1;
                if baud96rx = getbit then
                        tik <= '1';
                elsif baud96rx = baudclk then
                        baud96rx <= (others => '0');
                end if;
        end if;
end if;
end process;
--================================================
-- State Machine UART_Rx
--================================================
rxFSM: process(rst,clk,tik)
begin
if rst = '1' then
        akustate <= idle;
        modemout <= (others => '0');
        bitctr <= "000";
        write_en <= '0';
        dataout <= (others => '0');
elsif rising_edge(clk) then
        write_en <= '0';
        case akustate is
        when idle =>
        bitctr <= "000";
        if serin = '0' then
```

```
            akustate <= startdetect;
    else
            akustate <= idle;
    end if;
    when startdetect =>
    if tik = '1' then
            akustate <= getdata;
    else
            akustate <= startdetect;
    end if;
    when getdata =>
    if tik = '1' then
            modemout <= serin & modemout(7 downto 1);
            if bitctr = "111" then
                    akustate <= stopconfirm;
            else
                    bitctr <= bitctr + 1;
                    akustate <= getdata;
            end if;
    else
            akustate <= getdata;
            modemout <= modemout;
            bitctr <= bitctr;
    end if;
    when stopconfirm =>
    if tik = '1' then
            akustate <= idle;
            dataout <= modemout;
            if serin = '1' then
                    write_en <= '1';
            else
                    write_en <= '0';
```

```
                end if;
            else
                akustate <= stopconfirm;
            end if;
            when others =>
            akustate <= idle;
            end case;
    end if;
    end process;
    end Behavioral;
```

3.4.3 Perancangan modul VHDL Arcfour Cryptoprocessor

a. Algoritma arcfour

Kunci arcfour yang digunakan pada tugas akhir ini merujuk pada test vector dari CRYPTLIB (Internet Draft Kaukonen arcfour) yakni 0x01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef. Algoritma arcfour dideskripsikan dalam pseudocode dibawah ini.

```
keyschedule( )
{
 /* initialization */
 for i = 0 to 255
  s [i] = i;

 /* scrambling */.
 j = 0;
 for i = 0 to 255
 {
  j = (j + K [i % key_length] + s[i]) % 256;
  swap s[i] and s [j];
 }
}
pnrg ()
```

```
{
 i = 0;
 j = 0;
 while not end of stream
 {
  i = (i + 1) % 256;
  j = (j + s [i]) % 256;
  swap s[i] and s[j];
  t = s[i] + s[j];
  ct[i] = pt[i] xor s[t];
 }
}
```

b. Identifikasi Komponen

Komponen-komponen yang teridentifikasi adalah sebagai berikut ;

- 8 x 256 bit ROM sebagai kunci
- 8 x 256 bit dual port RAM sebagai S-Box
- 1 buah counter 8 bit sebagai pointer i
- 1 buah register 8 bit sebagai pointer j
- 1 buah blok kendali dalam bentuk FSM

c. Rekonstruksi Kode VHDL

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

entity arcfourrtl is
    port (
            clk: in STD_LOGIC;
            rst: in STD_LOGIC;
            enkrip : in STD_LOGIC;
            data_en: in STD_LOGIC;
            plainteks: in STD_LOGIC_VECTOR (7 downto 0);
```

```
        kamufull: out std_logic;
        cipherteks: out STD_LOGIC_VECTOR (7 downto 0)
    );
end arcfourrtl;

architecture arcfourrtl of arcfourrtl is
signal rc4_rdx: std_logic;
signal keyout: std_logic_vector(7 downto 0);
signal addrkey: std_logic_vector(7 downto 0);
type RAMtype is array (0 to 255) of std_logic_vector(7 downto 0);
shared variable arcRAM : RAMtype;
constant kunci : RAMtype :=
x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
```

```
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef",
    x"01",x"23",x"45",x"67",x"89",x"ab",x"cd",x"ef");
signal wea: std_logic;
signal web: std_logic;
signal doa: std_logic_vector(7 downto 0);
signal dob: std_logic_vector(7 downto 0);
signal addra: std_logic_vector(7 downto 0);
signal addrb: std_logic_vector(7 downto 0);
signal dia: std_logic_vector(7 downto 0);
signal dib: std_logic_vector(7 downto 0);
signal tahap: std_logic_vector(2 downto 0);
signal regJ: std_logic_vector(7 downto 0);
signal addr1_sig: std_logic_vector(7 downto 0);
signal ctr8bit: std_logic_vector(7 downto 0);
signal sumsbox: std_logic_vector(7 downto 0);
signal scram_done: std_logic;
signal order: std_logic_vector(1 downto 0);
signal goctr: std_logic;
signal addrcode: std_logic_vector(1 downto 0);
signal rdydone: std_logic;
signal cip_sig: std_logic_vector(7 downto 0);
```

```
signal cip_key: std_logic_vector(7 downto 0);
signal nextstep: std_logic;
begin
--=============================================
-- rom key
keyout <= kunci(conv_integer(addrkey));
--=============================================
-- ram Sbox
    process (CLK,WEA)
  begin
    if CLK'event and CLK = '1' then
                    if WEA = '1' then
          arcRAM(conv_integer(ADDRA)) := DIA;
       end if;
       DOA <= arcRAM(conv_integer(ADDRA));
    end if;
  end process;


    process (CLK,WEB)
  begin
    if CLK'event and CLK = '1' then
                    if WEB = '1' then
                            arcRAM(conv_integer(ADDRB)) := DIB;
       end if;
       DOB <= arcRAM(conv_integer(ADDRB));
    end if;
  end process;
--=============================================
--rc4 control
--ctrdone <= ctr8bit(7) and ctr8bit(6) and ctr8bit(5) and ctr8bit(4) and
ctr8bit(3) and ctr8bit(2) and ctr8bit(1) and ctr8bit(0);
order(1) <= tahap(2);
```

```
order(0) <= (not tahap(1) and tahap(0)) or (not tahap(2) and tahap(1));
addrcode(1) <= tahap(1) and  tahap(0);
addrcode(0) <= (tahap(1) xor tahap(0)) or (tahap(2) or scram_done);
addra <= "00000000" when order = "00" else
          sumsbox when order = "11" else
          ctr8bit + scram_done;
dia <= ctr8bit when order = "01" else
        dob;
addr1_sig <= "00000000" when addrcode = "00" else
                regJ + keyout + doa when addrcode = "10" else
                regJ + doa when addrcode = "11" else
                regJ;
wea <= (not tahap(1)) and (tahap(2) xor tahap(0));
web <= tahap(2) and (tahap(1) nor tahap(0));
sumsbox <= doa + dob;
addrb <= addr1_sig;
dib <= doa;
addrkey <= ctr8bit;
nextstep <= data_en and enkrip;
cipherteks <= cip_sig;
rc4_rdy <= not rc4_rdx;
--============================================
-- counter
process(rst,goctr,clk)
begin
    if rst = '1' then
        ctr8bit <= "00000000";
    elsif goctr = '1' then
        if rising_edge(clk) then
            ctr8bit <= ctr8bit + 1;
        end if;
    end if;
```

```
end process;
--================================================
-- FSM
process(rst,clk,data_en,enkrip,ctr8bit,scram_done)
begin
    if rst = '1' then
            tahap <= "000"; -- idle
            kamufull <= '0';
            cip_key <= "00000000";
    elsif rising_edge(clk) then
            kamufull <= rc4_rdx and data_en;
            rdydone <= rc4_rdx;
            if rc4_rdx = '1' and rdydone = '0' then
                    cip_key <= doa;
            else
                    cip_key <= cip_key;
            end if;
            if data_en = '1' then
                    if enkrip = '1' then
                            cip_sig <= plainteks xor cip_key;
                    elsif enkrip = '0' then
                            cip_sig <= plainteks;
                    end if;
            else
                    cip_sig <= cip_sig;
            end if;
            case tahap is
                    when "000" =>           -- idle
                    scram_done <= '0';
                    regJ <= "00000000";
                    rc4_rdx <= '0';
                    goctr <= '1';
```

```
tahap <= "001";
when "001" =>          -- init
if ctr8bit = "11111111" then
        goctr <= '0';
        tahap <= "010";
else
        tahap <= "001";
        goctr <= '1';
end if;
when "010" =>          -- read i
goctr <= '0';
tahap <= "011";
when "011" =>          -- read j
goctr <= '1';
regJ <= addr1_sig;
tahap <= "100";
when "100" =>          -- swap ij
if scram_done = '0' then
        if ctr8bit = "11111111" then
                goctr <= '0';
                scram_done <= '1';
                regJ <= "00000000";
        else
                goctr <= '0';
                tahap <= "010";
        end if;
else
        goctr <= '0';
        tahap <= "101";
end if;
```

```
                when "101" =>           --      readsum
                rc4_rdx <= '1';
                tahap <= "110";
                when "110" =>           -- encrypt
                if nextstep = '1' then  -- data_en = '1' then
                        goctr <= '0';
                        rc4_rdx <= '0';
                        tahap <= "010";         --read i
                else
                        tahap <= "110";
                end if;
                when others => tahap <="000";
            end case;
        end if;
    end process;
    end arcfourrtl;
```

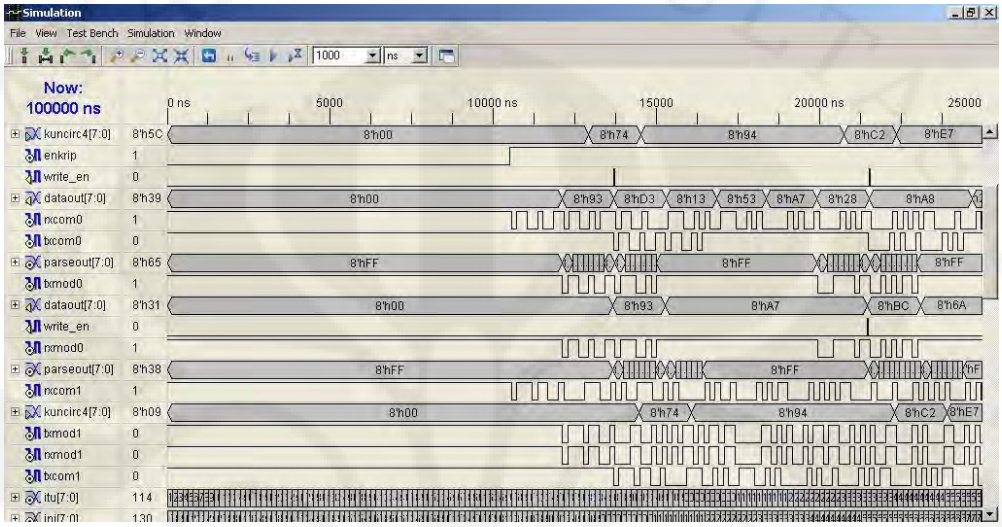
d. Simulasi Fungsional

Untuk simulasi fungsional menggunakan aplikasi Xilinx ISE simulator dengan VHDL testbench sebagai simulasi data yang dikirim oleh komputer dan modem melalui protokol UART. Pada komputer data dimasukkan melalui aplikasi Hyperterminal dengan baudrate 57600 bps, begitu juga dengan modem menggunakan baudrate 57600 bps, tetapi untuk simulasi fungsional baudrate yang digunakan pada VHDL testbench adalah 6,25 Mbps dengan clock system FPGA 100 MHz.

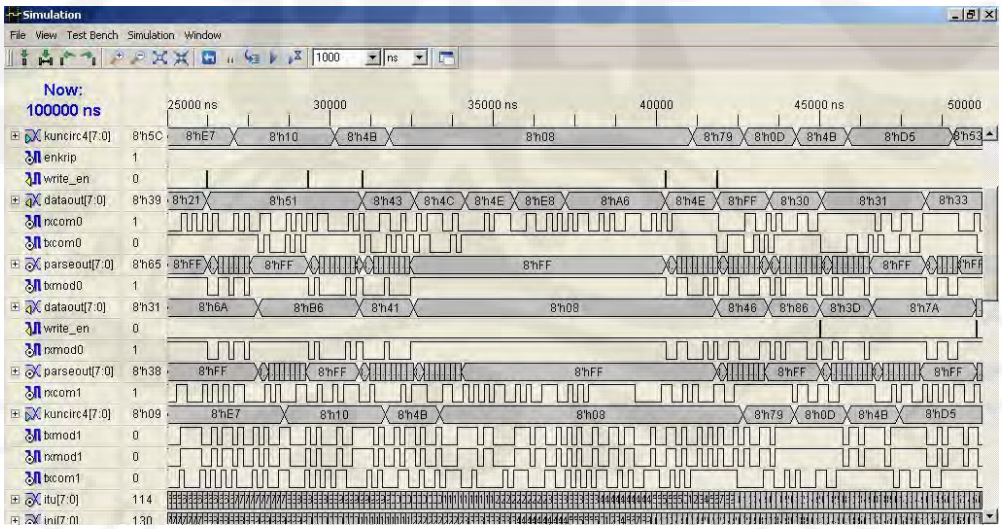
Jika pada VHDL testbench tetap menggunakan baudrate 57600 bps dan clock system FPGA 100 MHz, maka untuk mendapatkan nilai 1 bit data serial maka system harus menunggu 100 MHz/57600 ≈ 1763 clock (11 bit biner) yang setara dengan 17630 ns atau untuk menunggu 1 byte data dibutuhkan waktu 176300 ns.

Untuk menyingkat waktu simulasi dan karena keterbatasan memory pada komputer yang digunakan untuk simulasi sekaligus untuk menguji sistem jika digunakan untuk baudrate yang berbeda maka

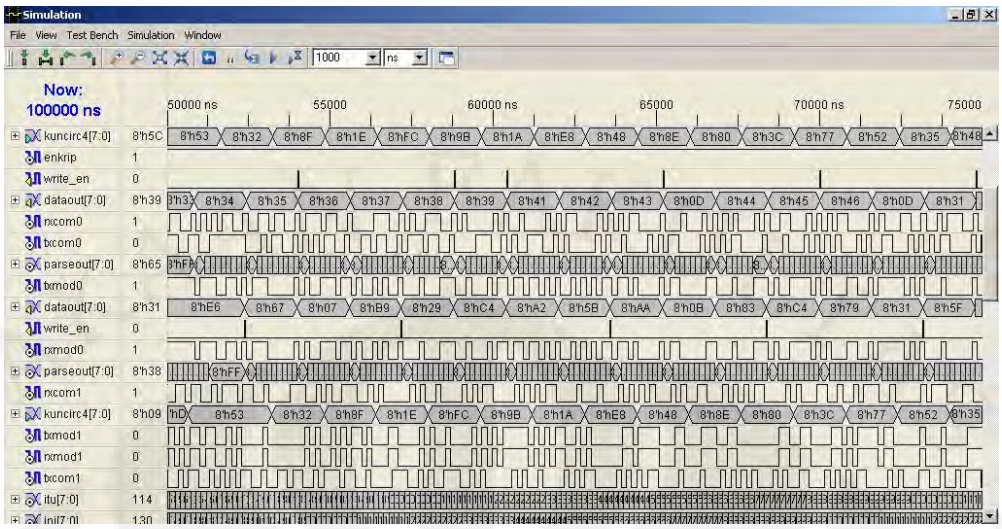
baudrate yang digunakan untuk simulasi adalah 6,25 Mbps, yang diperoleh dari 100 MHz/16 clock (4 bit biner) = 6,25 Mbps, sehingga waktu yang dibutuhkan untuk mendapatkan 1 bit data serial adalah 160ns.



Gambar 3.9 Simulasi fungsional 0 s/d 25000 ns



Gambar 3.10 Simulasi fungsional 25.000 s/d 50.000 ns

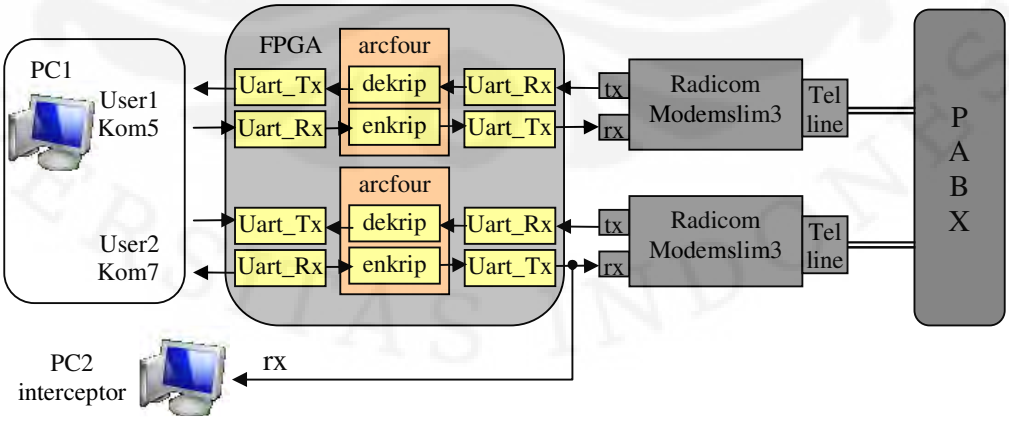


Gambar 3.11 Simulasi fungsional 50.000 s/d 75.000 ns

3.5 Implementasi Rancangan

3.5.1 Skema Rancangan

Karena keterbatasan perangkat yang tersedia modul kripto baik sisi pengirim maupun sisi penerima akan dimplementasikan pada 1 (satu) FPGA Xilinx Spartan-IIE LC Development Board, antarmuka pengguna juga hanya menggunakan 1 (satu) perangkat komputer personal dengan membuka 2 (dua) jendela aplikasi Hyperterminal Private Edition secara simultan dan penambahan 1 (satu) perangkat personal sebagai "*Interceptor*", data yang diterima oleh *interceptor* adalah data 'rx' pada salah satu modem. Maka skema rancangan pada gambar 3.1 mengalami perubahan menjadi seperti gambar 3.12 dibawah ini.



Gambar 3.12 Skema Rancangan Implementasi

3.5.2 HDL Synthesis

No	Komponen dasar yang diidentifikasi oleh XST	Jumlah
1	256x8-bit dual-port block RAM	4
2	256x8-bit ROM	4
3	4x1-bit ROM	8
4	11 bit adder	8
5	3 bit adder	8
6	8 bit adder	16
7	8-bit up counter	4
8	1-bit register	36
9	11-bit register	8
10	3-bit register	8
11	8-bit register	24
12	11-bit 4-to-1 multiplexer	8
13	8-bit 4-to-1 multiplexer	8
14	1-bit xor2	8
15	8-bit xor2	4

Tabel 3.1 HDL Synthesis Report

3.5.3 Advance HDL Synthesis

No	Komponen dasar yang diidentifikasi oleh XST	Jumlah
1	FSM	3
2	256x8-bit dual-port block RAM	4
3	256x8-bit ROM	4
4	4x1-bit ROM	8
5	11 bit adder	8
6	3 bit adder	8
7	8 bit adder	16
8	8-bit up counter	4
9	Flip-Flop	92

10	11-bit 4-to-1 multiplexer	
11	8-bit 4-to-1 multiplexer	
12	1-bit xor2	8
13	8-bit xor2	8

Tabel 3.2 Advance HDL Synthesis Report

3.5.4 Low Level Synthesis

Pemanfaatan Cell	Jumlah
BELS	1081
AND2	3
BUF	3
GND	1
INV	13
LUT1	32
LUT2	84
LUT2_D	4
LUT2_L	8
LUT3	180
LUT3_D	16
LUT3_L	24
LUT4	396
LUT4_D	20
LUT4_L	36
MUXCY	112
MUXF5	36
VCC	1
XORCY	112
FlipFlops/Latches	404
FDC	160
FDCE	128
FDE	80

FDP	36
RAMS	4
RAMB4_S8_S8	4
Clock Buffers	1
BUFGP	1
IO Buffers	11
IBUF	6
OBUF	5

Tabel 3.3 HDL Low Level Synthesis Report

3.5.5 Rangkuman Synthesis

Pemanfaatan Device FPGA	Jumlah	persentase
Number of Slices	440 out of 3072	14%
Number of Slice Flip Flops	404 out of 6144	6%
Number of 4 input LUTs	813 out of 6144	13%
Number of IOs	12	
Number of bonded IOBs	12 out of 329	3%
Number of BRAMs	4 out of 16	25%
Number of GCLKs	1 out of 4	25%
Actual Ratio	15	
Level of Logic	12	
Maximum frequency	76.970MHz	

Tabel 3.4 Design Summary Synthesis Report

BAB 4
PENGUJIAN DAN ANALISIS

4.1 Pengujian

4.1.1 Verifikasi Formal

Verifikasi formal dilakukan dengan membandingkan nilai pada hasil implementasi dengan test vektor yang digunakan.

Test vektor dari Cryptlib :

Plain Text: 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Key: 0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF

Cipher Text: 0x74, 0x94, 0xC2, 0xE7, 0x10, 0x4B, 0x08, 0x79

Hasil pengujian :

Gambar 4.1 Hasil Pengujian Rangkaian kunci arcfour

4.1.2 Ujicoba Operasional Kirim Terima file

No	Nama file	Type file	Ukuran file	Keterangan
1	Ilusi.gif	Grafis	19 kb	Berhasil
2	Yamahafz.jpg	Grafis	45 kb	Berhasil
3	Classic car.bmp	Grafis	181 kb	Gagal sinkonisasi
4	ez2susb_ds.pdf	document	99 kb	Putus transmisi
5	radicom modem.JPG	Grafis	25 kb	Aplikasi Hang
6	HPE ReadMe.DOC	Document	24kb	berhasil
7	seminario advanced.swf	Audio video	317 kb	gagal
8	Al-mulk.mp3	Audio	3076 kb	FPGA error

Tabel 4.1 Hasil Ujicoba kirim terima file

4.2 Analisis

4.2.1 Rekonstruksi Kode VHDL

1. Penulisan kode VHDL lebih cenderung menggunakan teknik deskripsi Register Transfer Level (RTL) pada setiap modul yang dirancang dan menggunakan teknik struktur untuk merancang arsitektur rancangan.
2. Tidak adanya modul VHDL untuk protokol komunikasi membuat rancangan mengalir tanpa adanya kendali atau deteksi kesalahan data yang dikirim.

4.2.2 Simulasi Fungsional

Simulasi fungsional yang dilakukan dengan Xilinx ISE Simulator lebih memberikan gambaran lingkungan yang sangat ideal, terlebih lagi jika terjadi kesalahan data yang nilainya di luar nilai ‘0’ dan ‘1’. Sehingga walaupun secara simulasi fungsional sistem telah memenuhi spesifikasi yang diinginkan, tetapi masih ada celah untuk kesalahan yang terjadi pada saat emulasi perangkat keras.

4.2.3 Implementasi Rancangan

Berdasarkan hasil laporan proses synthesis rancangan yang dibuat telah memenuhi parameter-parameter perancangan VHDL yang sesuai dengan

proses sysnthesis sehingga rancangan berhasil dimplementasikan ke dalam Spartan-IIE LC Development Board.

4.2.4 Verifikasi Formal

Pada verifikasi formal rancangan sudah sesuai dengan spesifikasi algoritma enkripsi arcfour sesuai dengan test vector.

4.2.5 Ujicoba Operasional Kirim Terima File

Kegagalan dalam proses kirim terima file lebih disebabkan karena tidak adanya modul protokol komunikasi dan protokol kriptografi. Algoritma enkripsi arcfour merupakan algoritma kriptografi stream cipher yang berarti rangkaian kunci acak yang digunakan oleh proses enkripsi dan dekripsi harus simetrik.

Jika pada kirim terima karakter teks yang panjang kesalahan yang terjadi mungkin hanya terjadi pada beberapa karakter saja, tetapi pada proses kirim terima file aplikasi hyperterminal menggunakan Zmodem Protocol. Zmodem protocol membuat paket-paket data dan fungsi *hash* pada data-data yang dikirimkan, maka apabila terjadi satu byte saja data ada yang rusak dan tidak dapat diperbaiki oleh error correction yang dimiliki maka sinkronisasi untuk proses kirim terima file menjadi gagal dan file tidak dapat dikirimkan.

BAB 5
KESIMPULAN

Dalam bab ini akan disimpulkan beberapa hal yang telah dibahas pada bab-bab sebelumnya serta hasil-hasil yang telah diperoleh dari pengujian dan ujicoba :

a. Implementasi algoritma enkripsi arcfour dan modul interface UART berhasil dilakukan pada FPGA Xilinx Spartan-IIE LC Development Board menggunakan bahasa deskripsi perangkat keras VHDL.
b. Modul kripto yang diimplementasikan berhasil melakukan enkripsi dan dekripsi pada kirim-terima data berupa teks, tetapi untuk kirim-terima data berupa file masih belum sempurna.
c. Kegagalan yang terjadi pada ujicoba operasional kirim terima file dimungkinkan karena perbedaan parameter elektrik pada I/O header FPGA, Radicom Modemslim3 dan kabel USB to Serial TTL, sehingga perlu dilakukan penelitian lebih lanjut.

DAFTAR REFERENSI

1. Pedroni, Volnei A (2004), *Circuit Design with VHDL,* MIT Press Cambridge, Massachusetts, London, England.
2. Pong P Chu (2006), *RTL Hardware Design Using VHDL*, John Wiley & Sons, Inc, Hoboken, New Jersey
3. Pong P Chu (2008), *FPGA Prototyping By VHDL Examples*, John Wiley & Sons, Inc, Hoboken, New Jersey
4. *Perancangan Rangkaian Enkripsi Algoritma RC4*. (Laboratorium IC Design, Pusat Mikroelektronika, ITB dalam Training Lembaga Sandi Negara, 2007)
5. Modul Pelatihan VHDL Lembaga Sandi Negara. (System Electronics Laboratory STTTelkom, 2004)
6. Kaukonen,Thayer, *A Stream Cipher Encryption Algorithm Arcfour,* Internet Draft, http://www.ietf.org/ietf/1id-abstracts.txt, 1999.
7. "*What are* FPGAs?". http://www.fpga4fun.com/fpgainfo1, 2009.
8. "Field Programmable Gate array", http://en.wikipedia.org/wiki/Field-programmable.gate.array, 2009.
9. Leibson's Law, "Moving Toward a New SOC System-Design-Methodology", http://www.edn.com/blog/980000298/post/500041850.html, 2009
10. Spartan-IIE LC Development Board User's Guide version 1.2. (MEMEC Design, 2004)
11. Parno, "Data Flow Diagram", Lecture notes: Sistem Informasi, www.scribd.com/doc/41618505/SI-03-DFD , 2010
12. "RC4", http://en.wikipedia.org/wiki/rc4, 2009.
13. "VHDL (VHSIC Hardware Description Language)", http://en.wikipedia.org/wiki/VHDL 2009.
14. "Xilinx Spartan-II Family Architecture", www.xilinx.com, 2004
15. Anantha P. Chandrakasan, and Donald E. Troxel, "3D FPGA Design and CAD Flow, chapter 2", MTL/RLE Groups, 2006
16. Syahral, Mohamad, Analisa Perbandingan Kode VHDL Dengan Menggunakan Teknik Deskripsi Perilaku Sistem Dan Teknik Deskripsi Aliran Data Pada Aplikasi Xilinx ISE Webpack, seminar, Universitas Indonesia 2010.