Perancangan dan Implementasi Algoritma DES untuk Mikroprosesor Enkripsi dan Dekripsi pada FPGA

Imaduddin Amrullah Muslim [1)] , R.Rizal Isnanto [2)], Eko Didik Widianto [3)]
Program Studi Sistem Komputer, Fakultas Teknik, Universitas Diponegoro
Jl. Prof.Sudharto, Tembalang, Semarang, Indonesia
Email: bangimad01@gmail.com

Abstrak— ***Seiring dengan semakin luasnya penerapan teknologi komputasi di sekitar kita, menjadikan informasi menjadi sangat mudah dan cepat untuk disebarkan. Kita dapat mengakses informasi dan data-data yang kita butuhkan dengan mudah. Namun permasalahan yang kita hadapi saat ini kerhasiaan informasi menjadi sangat riskan. Oleh karena itu Sistem keamanan merupakan hal penting yang perlu diperhatikan dalam mengembangkan suatu sistem komputer hal ini lah yang menjadikan enkripsi dan dekripsi data menjadi hal yang penting.***

Modul rancangan IP Core ini dirancang menggunakan aplikasi Xilinx ISE Design Suite 12.4. Kemudian rancangan IP Core ini diimplementasikan pada papan Xilinx FPGA Spartan-3E XC3S500E-4FG320C dari keluarga Xilinx FPGA Spartan-3E dengan 500K sistem gerbang. Verifikasi fungsional dari IP Core yang dirancang menggunakan testbench dan simulasi diagram pewaktuan menggunakan aplikasi Xilinx ISE Simulator. Tugas akhir ini ditujukan untuk mengembangkan IP Core yang mampu menjalankan fungsi enkripsi DES (Data Encryption Standard) dan ditulis menggunakan bahasa Verilog. Implementasi algoritma enkripsi dan dekripsi algoritma DES telah berhasil dilakukan. Hasil analisis menunjukkan sistem telah dapat melakukan enkripsi dan dekripsi data sesuai dengan spesifikasi algoritma DES. Sebaiknya penelitian ini dikembangkan kembali dengan menguji sistem untuk melakukan transmisi data berupa file ataupun teks. Selain itu juga perlu meningkatkan unjuk kerja sistem dengan optimasi sumber daya dan kecepatan waktu proses pada perancangan rekonstruksi kode verilog.

Kata kunci— IP Core, Enkripsi, DES, Verilog

I. PENDAHULUAN

Perkembangan Teknologi Informasi yang begitu pesat hari ini sangatlah membantu kehidupan manusia. Teknologi informasi saat ini semakin canggih, tuntutan akan kemudahan mencari informasi menjadi faktor penggerak pesatnya pertumbuhan teknologi informasi. Selain itu, perkembangan teknologi internet juga tumbuh cukup pesat, hal ini mendukung untuk memudahkan masyarakat dalam mengakses informasi yang diinginkan.

Pemanfaatan Teknologi Informasi telah terbukti memudahkan kegiatan manusia untuk mengakses informasi dalam waktu yang singkat akan tetapi dikarenakan akses yang begitu luas, informasi atau data yang rahasia menjadi sangat riskan dapat diketahui oleh orang yang tidak berhak. Pada layanan komunikasi publik sangat rentan terhadap ancaman pihak ketiga untuk menyadap, merusak dan mengubah informasi yang dikirimkan.

Oleh karena itu tindakan pengamanan data mutlak diperlukan, tindakan yang dilakukan biasanya dengan melakuakan proses penyandian data yang biasa disebut enkripsi data. Metode ini mengkodekan suatu informasi sedemikian rupa dengan algoritma kriptografi tertentu sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak berhak mengetahuinya.

Proses enkripsi data dapat diimplementasikan dengan menggunakan suatu rangkaian fungsi logika sistem digital. Perancanagan suatu sistem dijital menggunakan konsep penggunaan kembali dapat meningkatkan produktivitas dari perancang.Melalui penggunaan konsep perancangan dengan menggunakan IP (*Intellectual Property*) *core*, seorang perancang dapat melakukan verifikasi dan simulasi terhadap rancangan yang telah dibuatnya. Dengan demikian dapat meminimalkan kesalahan yang terjadi sebelum proses produksi dilakukan.

Intellectual Property (IP) dalam bentuk fungsi-fungsi yang telah didefinisikan atau disebut juga *core*, saat ini telah menjadi topik pembahasan yang hangat dalam dunia elektronik [1]. Salah satu penerapan IP *core* adalah dalam mewujudkan suatu mikroprosesor ataupun mikrokontroler. Mikroprosesor yang dirancang dapat berupa mikroposesor dengan kebutuhan ataupun mikroprosesor dengan kebutuhan khusus. Penggunaan IP *core* dimulai dari automasi peralatan industri, alat komunikasi hingga mainan anak-anak.

Implementasi algoritma kriptografi dalam bentuk perangkat lunak sudah banya kita jumpai pada layanan yang disediakan oleh kalangan perbankan, seperti transaksi *online*, sms banking dan internet banking.Sedangkan implementasi algoritma kriptografi pada perangkat keras masih sedikit kita jumpai.Oleh karena itu tugas akhir ini menitikberatkan pada implementasi algoritma kriptografi pada perangkat keras untuk menjamin keras untuk menjaga kerahasiaan informasi yang dikomunikasikan.

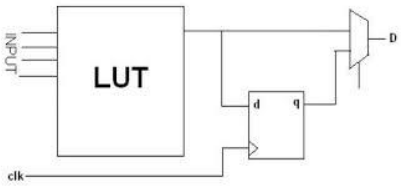
II. LANDASAN TEORI

A. 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).[2]

Karakteristik dari FPGA antara lain adalah dapat dirancang sesuai dengan keinginan dan kebutuhan pemakai

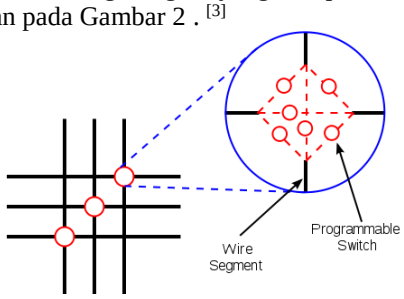
tanpa melalui tahap *burn* di laboratorium atau di-*hardwire* oleh perusahaan piranti, hal tersebut mungkin dilakukan karena FPGA terdiri atas sekumpulan *Configurable Logic Blocks* (CLB) yang terhubung melalui Programmable Interconnects (PI). Pada umumnya setiap CLB terdiri dari beberapa Logic Cells yang kadang disebut juga Slice, yang masing-masing tediri dari 4-input Look up Table (LUT), D Flip-Flop dan 2-to-1 Mux, seperti pada Gambar 1 dibawah ini.



Gambar 1 Logic cell pada FPGA

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 CLB ini.

Setiap Logic Cell dapat dihubungkan dengan Logic Cell lainnya melalui Programmable Interconnect (PI) yang akan membentuk suatu fungsi logika yang kompleks, ilustrasi dari PI ditunjukkan pada Gambar 2 . [3]



Gambar 2 Programmable Interconnects.

B. Perancangan sistem dengan Verilog

Perancangan sistem dengan Verilog menggunakan pendekatan secara top down, tahap-tahap perancangannya ditunjukkan dengan diagram alir pada Gambar 2.5.[6]. Diawali dengan analisa spesifikasi dari algoritma yang akan digunakan. Selanjutnya menentukan operasi yang dilakukan untuk menjalankan proses keseluruhan sistem. Kemudian seluruh operasi yang dilakukan dideskripsikan dalam bahasa deskripsi Verilog. Setelah rancangan operasi selesai maka dilakukan simulasi fungsional untuk mengetahui modul rancangan telah melakukan fungsi sesuai dengan yang diharapkan. Jika seluruh operasi yang dirancangan sudah menjalankan fungsinya dengan baik selanjutnya modul rancangan diimplementasikan dengan perangkat lunak sesuai dengan tipe dan jenis FPGA.

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) kategori 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 bertugasuntuk transfer/perpindahan data, terutama perpindahan data dari dan keluar sistem. Komponen ini merupakan modul antarmuka sistem yangakan berhubungan dengan perangkat lain di luar sistem.

2. Identifikasi Operasi Datapath

Melakukan identifikasi terhadap komponen-komponen yang telah diuraikan pada tahap sebelumnya, komponen apa saja yang merupakan komponen yang sudah standar (seperti full adder , decoder, multiplexerdan 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.

3. Rekonstruksi kode Verilog

Dalam rekonstruksi kode Verilog 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.

4. 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.

5. Implementasi Desain

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

C. Algoritma DES

DES termasuk sistem kriptografi simetri dan tergolong jenis blok kode. DES beroperasi pada ukuran blok 64 bit. DES mengenkripsi 64 bit teks-asli menjadi 63 bit teks-kode dengan 56 bit kunci internal (*internal key*) atau upa-kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*)

yang panjangnya 64 bit. Skema global dari algoritma DES sebagai berikut :

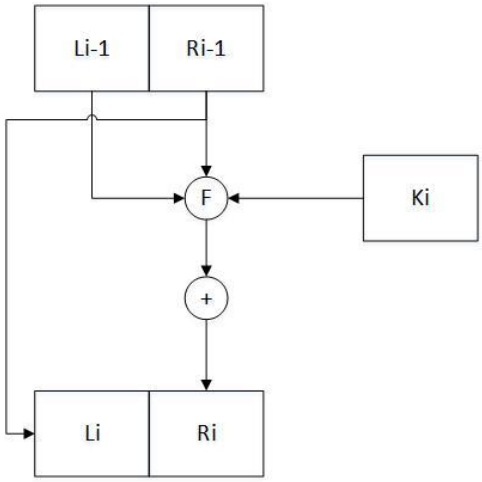
1. Blok teks-asli dipermutasi dengan matriks permutasi awal (*initial permutation* atau IP). Bisa ditulis x_0 = IP (x) = L_0R_0, di maba L_0 terdiri dari 32 bit pertama dari x_0 dan 32 bit terakhir dari R_0.
2. Hasil permutasi awal kemudian di-*enciphering* sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda dengan perhitungan LiRi.

$1 \le i \le 16$, dengan mengikuti aturan berikut:

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$ (1)

Dimana $\oplus$ merupakan exclusive-or dari dua. f adalah suatu fungsi dan K1, K2, . . ., K16 dengan panjang 48 dari perhitungan fungsi dari kunci K. (Sebenarnya Ki adalah permutasi dari K). K1, K2, . . . , K16 terdiri dari kunci skedul. Putaran pertama dari enkripsi tersebut ditunjukkan oleh Gambar 3.



Gambar 3 Skema satu putaran DES

3. Hasil enciphering kemudian dipermutasi dengan matriks permutasi balik (*invers initial permutation* atau IP^{-1}) menjadi blok teks-kode. IP^{-1}ke bitstring $R_{16}L_{16}$, memperoleh teks-kode y, kemudian $y=IP^{-1}(R_{16}L_{16})$.

Dalam proses *enciphering*, blok plainteks terbagu menjadi dua bagian, kiri (*L*) dan kanan (*R*), yang masing-masing panjangnya 32 bit. Pada setiap putaran *i*, blok*R* merupakan masukan untuk fungsi transformasi yang disebut *f*. Pada fungsi *f*, blok *R* dikombinasikan dengan kunci internal *Ki*. Keluaran dari fungsi *f* di-*XOR*-kan dengan blok*L* untuk mendapatkan blok *R* yang baru. Sedangkan blok*L* yang baru diambil dari blok *R* sebelumnya. Ini adalah satu putaran DES.

III. PERANCANGAN DAN IMPLEMENTASI

A. *Perancangan Operasi pada Algoritma DES*

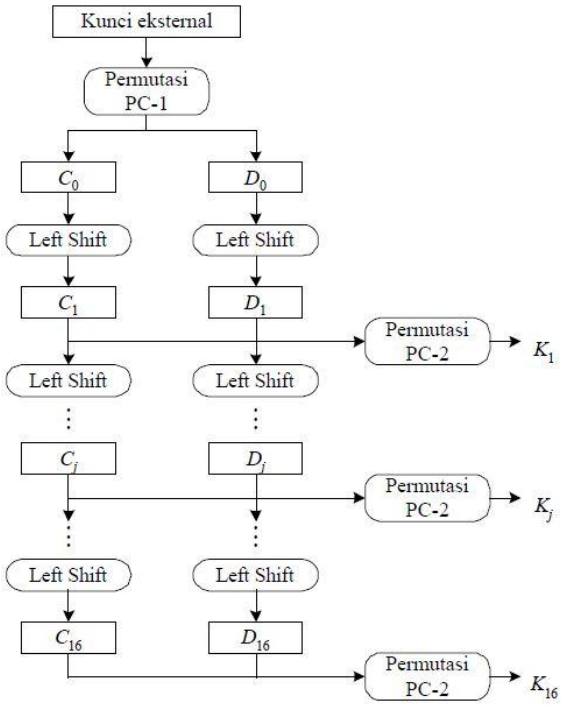
Rancangan mikroprosesor pada tugas akhir ini merupakan implementasi algoritma enkripsi DES ke dalam FPGA Xilinx Spartan3E sebagai modul kripto. Proses enkripsi menggunakan algoritma DES merupakan serangkaian operasi aritmatika dan logika yang mengubah posisi bit-bit dari *plainteks* sedemikian rupa sehingga pesan asli tidak dapat diketahui.Proses enkripsi ataupun dekripis melalui beberapa tahap operasi.

1. Permutasi awal

Operasi pertama yang dilakukan adalah memasukkan blok plainteks ke dalam permutasi awal (*initial permutation* atau IP). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan bit-bit didalamnya berubah. Pengacakan dilakukan dengan menggunakan matriks permutasi awal.

2. Pembangkitan kunci internal

Blok *enciphering* yang dilakukan pada algoritma DES berjumlah 16 putaran, maka dibutuhkan kunci internal sebanyak 16 buah, yaitu $K_1,K_2,\ldots,K_{16}$. Kunci-kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi. Secara umum proses pembangkitan kunci internal ditunjukkan pada Gambar 4.



Gambar 4 Proses pembangkitan kunci

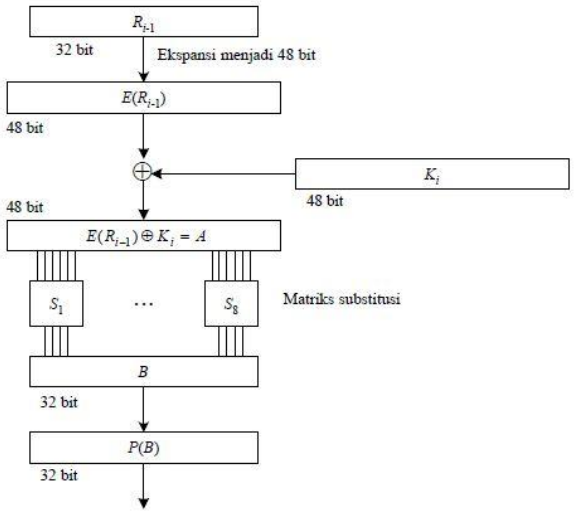
3. Enciphering

Proses enciphering terhadap blok plainteks dilakukan setelah permutasi awal. Setiap blok plainteks mengalami 16 kali putaran enciphering. Setiap putaran enciphering merupakan jaringan Feistel yang secara matematis dinyatakan sebagai

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

Diagram komputasi fungsi f diperlihatkan pada Gambar 5



Gambar 5 Diagram komputasi fungsi *f*

4. Permutasi akhir

Permutasi terakhir dilakukan setelah 16 kali putaran terhadap gabungan blok kiri dan blok kanan. Proses permutasi menggunakan matriks permutasi awal balikan (*inverse initial permutation* atau IP^{-1})

B. Implementasi Algoritma DES dengan Verilog

1. Implementasi modul permutasi awal

Modul verilog permutasi awal berfungsi untuk melakukan pengacakan posisi bit plainteks yang berjumlah 64 bit. Posisi bit diacak berdasarkan matriks permutasi awal. Lalu setelah diacak dengan matriks permutasi awal menghasilkan keluaran berupa 32 bit bagian kiri (L_0) dan 32 bit yang merupakan bit ke-33 sampai bit ke-64 dan bagian kanan (R_0) bit ke-1 sampai bit ke-32. Operasi pengacakan dengan matriks permutasi awal diatas jika sinyal chip_select_bar bernilai "0".

2. Implementasi modul permbangkitan kunci internal

Pada modul pembangkitan kunci internal ini mengubah kunci eksternal sepanjang 64-bit menjadi kunci internal yang berjumlah 16 buah yang akan digunakan pada tiap operasi jaringan Feistel yang berjumlah 16 putaran. Operasi terhadap kunci eksternal ini dengan melakukan permutasi dengan matriks permutasi kompresi PC-1. Setelah itu keluaran dari permutasi PC-1 dibagi dua yang akan menjadi masukan matriks PC-2 setelah dilakukan penggeseran kiri 1 bit atau 2 bit sesuai putaran yang dijelaskan pada proses pembangkitan kunci internal.

3. Implementasi modul enciphering

Modul enciphering ini berisi proses yang terjadi pada proses enciphering yang sudah dijelaskan pada sub bab 3.1.5 tentang operasi enciphering. Berupa komputasi fungsi *f* yang berupa fungsi ekspansi yang memperluas blok R_{i-1}. Lalu hasilnya di-XOR-kan dengan K_i yang didapatkan dari modul pembangkitan kunci internal. Setelah itu pemrosesan dengan 8 kotak-S yang kemudian keluarannya dipermutasi kembali dengan matriks permutasi *P* (*P-box*).

4. Implementasi modul permutasi akhir

Masukan pada operasi permutasi hasil keluaran proses enciphering yang posisinya sudah sesuai 32-bit kanan dan 32-bit kiri dengan matriks permutasi awal balikan.Menghasilkan keluaran berupa cipherteks sepanjang 64-bit.

5. Implementasi modul enkripsi

Modul yang menggabungkan seluruh proses enkripsi dari permutasi awal, pembangkitan kunci internal, 16 putaran proses enciphering hingga permutasi dengan matriks permutasi awal balikan digabungkan dalam satu modul untuk menyelaraskan semua operasi menjadi kesatuan proses enkripsi. Dimana masukannya berupa 64-bit plainteks dan 64-bit cipherteks sebagai keluaran.

6. Implementasi modul Dekripsi

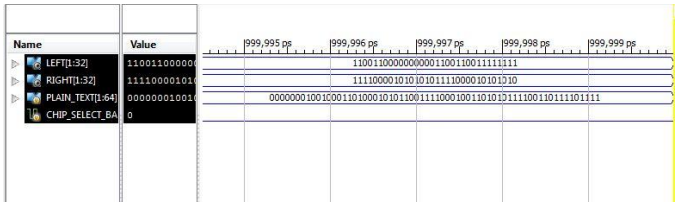
Modul yang menggabungkan seluruh proses dekripsi dari permutasi awal, pembangkitan kunci internal, 16 putaran proses deciphering hingga permutasi dengan matriks permutasi awal balikan digabungkan dalam satu modul untuk menyelaraskan semua operasi menjadi kesatuan proses dekripsi.

IV. HASIL DAN PEMBAHASAN

A. Simulasi modul rancangan

1. Simulasi modul permutasi awal

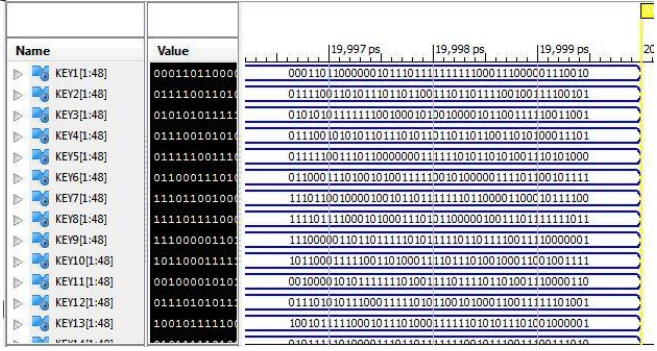
Setelah rancangan modul permutasi awal berhasil diimplementasikan pada FPGA Xilinx langkah selanjutnya adalah melakukan simulasi rangkaian menggunakan testbench dengan modul testbench yang sudah dirancang pada sub bab 3.2.1. Masukan yang dibutuhkan adalah berupa plainteks yang panjangnya 64-bit. Diharapkan keluaran berupa 32-bit kiri dan 32-bit kanan hasil permutasi dengan matriks permutasi awal. Hasil simulasi telah berhasil sesuai dengan keluaran yang diharapkan seperti yang ditampilkan pada diagram perwaktuan pada Gambar 6.



Gambar 6 Diagram pewaktuan modul permutasi awal

2. Simulasi modul permbangkitan kunci internal

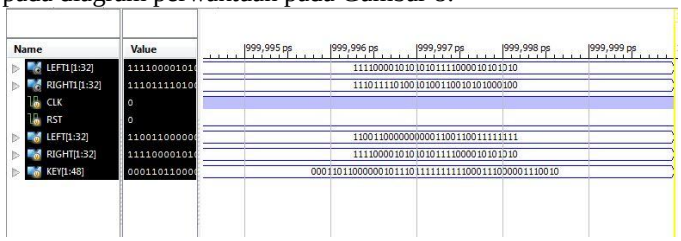
Setelah modul rancangan pembangkitan kunci internal berhasil diimplementasikan pada FPGA Xilinx langkah selanjutnya adalah melakukan simulasi rangkaian menggunakan testbench dengan modul testbench yang sudah dirancang. Masukan yang dibutuhkan adalah berupa kunci eksternal yang panjangnya 64-bit. Diharapkan 16 buah kunci internal yang panjangnya 48-bit yang merupakan hasil permutasi dengan matriks PC-1, pergeseran bit, dan permutasi dengan matriks PC-2. Hasil simulasi telah berhasil sesuai dengan keluaran yang diharapkan seperti yang ditampilkan pada diagram perwaktuan pada Gambar 7.



Gambar 7 Diagram modul pewaktuan pembangkitan kunci internal

3. Simulasi modul enciphering

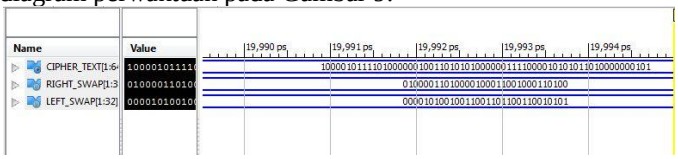
Setelah rancangan modul enciphering berhasil diimplementasikan pada FPGA Xilinx langkah selanjutnya adalah melakukan simulasi rangkaian menggunakan testbench dengan modul testbench yang sudah dirancang pada sub bab 3.2.3. Masukan yang dibutuhkan adalah berupa 32-bit kanan dan 32-bit kiri hasil keluaran modul permutasi awal. Diharapkan keluaran berupa 32-bit kiri dan 32-bit kanan hasil enciphering tiap putaran. Hasil simulasi telah berhasil sesuai dengan keluaran yang diharapkan seperti yang ditampilkan pada diagram perwaktuan pada Gambar 8.



Gambar 8 Diagram pewaktuan modul enciphering

4. Simulasi modul permutasi akhir

Setelah rancangan modul permutasi awal berhasil diimplementasikan pada FPGA Xilinx langkah selanjutnya adalah melakukan simulasi rangkaian menggunakan testbench dengan modul testbench yang sudah dirancang pad.. Masukan yang dibutuhkan adalah berupa 32-bit kanan dan 32-bit kiri hasil keluaran modul enciphering. Diharapkan keluaran berupa 64-bit cipherteks hasil permutasi dengan matriks permutasi awal balikan. Hasil simulasi telah berhasil sesuai dengan keluaran yang diharapkan seperti yang ditampilkan pada diagram perwaktuan pada Gambar 9.



Gambar 9 Diagram pewaktuan modul permutasi akhir

B. Implementasi rancangan pada FPGA

Modul algoritma DES yang telah dirancang ditulis dengan menggunakan bahasa HDL yaitu Verilog. Sebelum rancangan tersebut dapat diimplementasikan diatas papan Xilinx Spartan3E, rancangan modul akan disintesis dari HDL menjadi komponen-komponen dasar yang diidentifikasi oleh FPGA. Perangkat untuk sintesis dan verifikasi sudah tersedia pada aplikasi Xilinx ISE. Pada Tabel 1 menampilkan komponen-komponen yang dibutuhkan dan juga tersedia untuk mengimplementasikan modul rancangan di atas papan Xilinx Spartan3E.

Tabel 1 Laporan Sintesis HDL

No	Komponen dasar yang diidentifikasi oleh XST	Jumlah
1	64x4-bit ROM	128
2	32-bit Register	32
3	1-bit tristate buffer	120
4	32-bit xor2	16
5	48-bit xor2	16

Setelah sintesis komponen yang dibutuhkan untuk melakukan operasi rancangan maka FPGA akan melakukan sintesis low level. Untuk mengetahui cell apa saja yang dibutuhkan untuk mewujudkan komponen yang dibutuhkan dalam rancangan. Pada Tabel 2 menunjukkan hasil sintesis pada low level.

Tabel 2 Laporan sintesis Low level

No	Pemanfaatan Cell	Jumlah
1	BUF	9
2	LUT2	376
3	LUT2_D	88
4	LUT3	456
5	LUT3_D	21
6	LUT4	4985
7	LUT4_L	16
8	MUXF5	1009
9	MUXF6	32
10	FDC	1136
11	BUFGP	1
12	IBUF	67
13	OBUF	64

Selanjutnya setelah mendapatkan sintesis komponen apa saja yang dibutuhkan untuk mewujudkan modul yang dirancang. Selanjutnya aplikasi akan memberikan laporan berupa berapa banyak sumber daya komponen yang digunakan untuk mengimplementasikan rancangan. Jumlah komponen yang dibutuhkan dibandingkan dengan jumlah yang tersedia dalam persentase.

Tabel 3 Laporan rangkuman sintesis rancangan

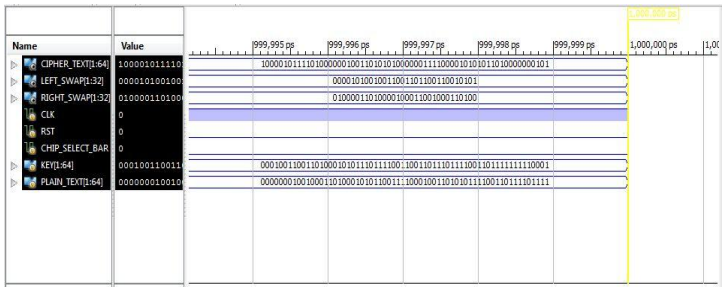
Pemanfaatan Device FPGA	Jumlah	Presentase
Number of Slice Flip Flops	1136 dari 9312	12%
Number of 4 input LUTs	5951 dari 9312	63%
Number of occupied Slices	3380 dari 4656	72%
Number of Slices containing only related logic	3380 dari 3380	100%
Number of Slices containing unrelated logic	0 dari 3380	0%
Total number of 4 input LUTs	5951 dari 9312	63%

Number of bonded IOBs	132 dari 232	56%
Number of BUFGMUXs	1 dari 24	4%
Average Fanout of Non-Clock Nets	4,40	

C. Analisis simulasi modul enkripsi dan dekripsi Algoritma DES

1. Simulasi modul Enkripsi

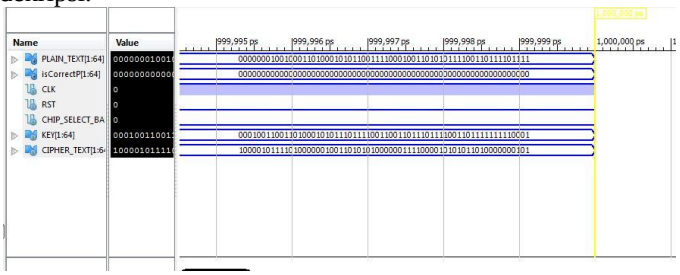
Simulasi fungsional terhadap rancangan yang telah dibuat menggunakan aplikasi Xilinx ISE simulator dengan Verilog testbench sebagai simulasi data yang dikirim. Bit data yang menjadi masukan plainteks untuk modul enkripsi dan kunci eksternal yang semuanya panjangnya 64-bit. Setelah itu keluaran akan ditampilkan dalam diagram pewaktuan. Operasi enkripsi akan berjalan jika sinyal chip_select_bar bernilai "0" dan sinyal reset bernilai "0". Sebagai inisiasi kita mengatur waktu tunda selama 10 nanodetik dan chip_select_bar dengan nilai "1". Maka jika dijalankan simulasi fungsional menggunakan diagram perwaktuan akan tampil seperti pada Gambar 10 untuk modul enkripsi.



Gambar 10 Diagram pewaktuan modul enkripsi

2. Simulasi modul Dekripsi

Simulasi fungsional terhadap rancangan modul dekripsi yang telah dibuat menggunakan aplikasi Xilinx ISE simulator dengan Verilog testbench sebagai simulasi data yang dikirim. Bit data yang menjadi masukan cipherteks untuk modul dekripsi dan kunci eksternal yang semuanya panjangnya 64-bit. Setelah itu keluaran akan ditampilkan dalam diagram pewaktuan. Operasi enkripsi akan berjalan jika sinyal chip_select_bar bernilai "0" dan sinyal reset bernilai "0". Sebagai inisiasi kita mengatur waktu tunda selama 10 nanodetik dan chip_select_bar dengan nilai "1". Maka jika dijalankan simulasi fungsional menggunakan diagram perwaktuan akan tampil seperti pada Gambar 11 untuk modul dekripsi.



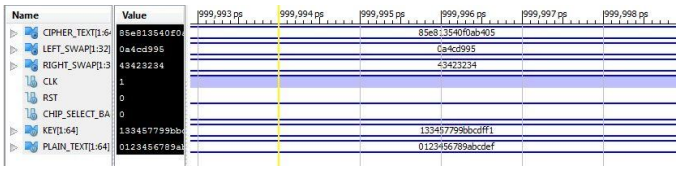
Gambar 11 Diagram pewaktuan modul dekripsi

3. Analisa modul enkripsi dan Dekripsi

a. Rekonstruksi kode verilog : Penulisan kode Verilog lebih cenderung menggunakan teknik struktur pada setiap modul yang dirancang.

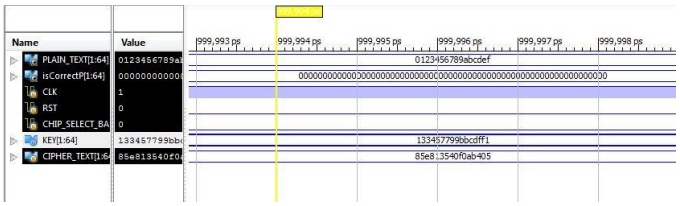
b. Implementasi Rancangan : Berdasarkan hasil laporan proses sintesis rancangan yang dibuat telah memenuhi parameter-parameter perancangan Verilog yang sesuai dengan proses sintesis. Sehingga rancangan berhasil diimplementasikan ke dalam Xilinx Spartan3E.

c. Verifikasi fungsional rancangan : Verifikasi fungsional ini dilakukan dengan melakukan enkripsi sebuah plainteks dengan panjang 64-bit pada modul enkripsi. Keluaran modul enkripsi berupa cipherteks yang panjangnya juga 64-bit. Cipherteks keluaran dari modul enkripsi menjadi masukan pada modul dekripsi dan diharapkan keluaran dari modul dekripsi berupa plainteks yang sama dengan masukan pada modul enkripsi. Plainteks yang menjadi masukan pada modul enkripsi adalah '0123456789ABCDEF' yang ditulis dalam heksa desimal. Dengan menggunakan kunci sepanjang 64-bit yaitu '133457799BBCDFF1' dalam heksa decimal. Menghasilkan keluaran berupa cipherteks yaitu '85E813540F0AB405'. Pada verifikasi fungsional rancangan sudah sesuai dengan spesifikasi algoritma enkripsi DES seperti yang ditunjukkan pada Gambar 12



Gambar 12 Verifikasi fungsional enkripsi

Selanjutnya keluaran modul enkripsi berupa cipherteks menjadi masukan modul dekripsi. Dengan menggunakan kunci yang sama diharapkan menghasil plainteks yang sama dengan masukan pada modul enkripsi yaitu '0123456789ABCDEF'. Verifikasi fungsional rancangan sudah sesuai dengan spesifikasi algoritma dekripsi DES seperti yang ditunjukkan pada Gambar 13.



Gambar 13 Verifikasi fungsional dekripsi

V. KESIMPULAN

A. Kesimpulan

Kesimpulan dari beberapa hal yang telah dibahas pada bab-bab sebelumnya serta hasil-hasil yang telah diperoleh dari simulasi:

1. Tugas akhir ini telah memberikan kontribusi IP*core* mikrokontroler enkripsi dan dekripsi menggunakan algoritma DES
2. Implementasi algoritma enkripsi dan dekripsi DES berhasil dilakukan pada FPGA Xilinx Spartan3E Development Board menggunakan bahasa deskripsi perangkat keras Verilog
3. Hasil perancangan modul enkripsi dan dekripsi telah disimulasikan dan diverifikasi melalui diagram perwaktuan dengan aplikasi Xilinx ISE Simulator. Hasil simulasi menunjukkan bahwa modul enkripsi dan dekripsi dapat bekerja dengan baik.

B. Saran

Berdasarkan pengujian terhadap aplikasi *quick count* yang telah dibuat, dapat diberikan beberapa saran sebagai berikut.

1. Rancangan modul enkripsi dan dekripsi menggunakan algoritma DES ini masih bisa dikembangkan dengan mengimplementasikan pada modul komunikasi atau pun transmisi data.
2. Penelitian sebaiknya melakukan pengujian dengan menambahkan modul untuk mengirim data baik berupa teks ataupun file.

DAFTAR PUSTAKA

[1] Surian, Didi., *Perancangan IPCore Mikrokontroler Kompatibel ATMEL ATMega8535 dengan VHDL*, Tesis S-2, Universitas Indonesia, Depok, 2008.

[2] Syahrial, M., *Perancangan dan Implementasi Algoritma Enkripsi Arcfour pada Perangkat Kriptografi berbasis FPGA*, Skripsi S-1, Universitas Indonesia, Depok, 2011.

[3] *What are* FPGAs? . http://www.fpga4fun.com/fpgainfo1, September 2014.

[4] *Leibson's Law, "Moving Toward a New SOC System-Design-Methodology",* http://www.edn.com/blog/980000298/post/500041850.html, September 2014

[5] *Xilinx Spartan3E Architecture*, www.xilinx.com, September 2014

[6] Pusat Mikroelektronika ITB, "Perancangan Rangkaian Enkripsi Algoritma RC4.", Lembaga Sandi Negara, Bandung, 2007.

[7] Ariyus, Doni., *Pengantar Ilmu Kriptografi-Teori Analisis dan Implementasi*, Andi, 2008.

()