PERANCANGAN DAN IMPLEMENTASI ARSITEKTUR *DEMAPPER* 64-QAM DI FPGA (*FIELD PROGRAMABLE GATE ARRAY*)

DESIGN AND IMPLEMENTATION OF ARCHITECTURE DEMAPPER 64-QAM ON FPGA (FIELD PROGRAMABLE GAT ARRAY)

Achmad Rizal Mauludin[1], Dr. Ir. Rina Pudji Astuti,MT[2], Denny Darlis, S.Si, M.T[3]
Fakultas Elektro dan Komunikasi Institut Teknologi Telkom, Bandung
achmad.rizal.mauludin@gmail.com[1], rpa@ittelkom.ac.id[2], dennydarlis@gmail.com[3]

Abstrak

Sistem telekomunikasi bertujuan untuk mengirimkan sinyal dari sumber informasi yang dapat berbentuk suara, pesan singkat atau Short Message Service (SMS), gambar, video dan layanan data ke tujuan yang diinginkan. Informasi yang akan dikirimkan akan diubah menjadi sinyal yang dapat dilewati media transmisi, dan agar sinyal yang diterima disisi penerima dapat dibaca, diperlukan *demodulator* yang dapat mengubah sinyal yang diterima menjadi informasi seperti yang dikirimkan. *Demodulator* 64-Quadrature Amplitude Modulation (QAM) adalah salah satu jenis *demodulator* yang mampu mendemodulasi sinyal frekuensi tinggi.

Dalam tugas akhir ini, telah dirancang dan diimplementasikan *demapper* 64-QAM yang merupakan sub blok *demodulator*, pada FPGA (Field Programable Gate Array) yang menggunakan bahasa pengkodean *Very High Speed Integrated Cicuit* (VHSIC*) Hardware Description Language* (VHDL) Fungsi dari blok ini adalah untuk memetakan balik simbol-simbol masukan dengan amplitudo dan fasa yang berbeda-beda yang sebelumnya telah direpresentasikan ke dalam bentuk bit-bit pada sisi pengirim. Pemetaan balik ini mengubah simbol-simbol tersebut menjadi bit-bit informasi yang masih berupa bit-bit *inphase* dan *quadrature*.

Dari hasil penelitian ini, untuk kondisi ideal atau gangguan didapatkan *output* di sisi penerima berupa sebuah bit-bit informasi yang sama dengan bit-bit informasi yang dikirimkan pada sisi pengirim. Sedangkan untuk kondisi ada gangguan, hasil outputnya masih sama dengan bit-bit informasi selama bit yang diganggu adalah enam bit dari LSB (*Least Significant bit*), untuk tujuh bit yang diganggu *error process* yang terjadi adalah 21,8310 % sedangkan untuk empat belas bit yang diganggu *error process* yang terjadi sebesar 96,9072%.

Kata kunci: 64-QAM, Demodulasi Digital, FPGA, VHDL.

Abstract

Telecommunication system purpose to transmit signal from information source / transmitter which is like voice, short message service, image, video and data service to the destination or receiver. The information who will be transmitted, will be changed into signal who can be passed by transmision medium, and in order that received signal can be read, we need a demodulator which is can change received signal into the information like a transmitted signal. Demodulator 64-Quadrature Amplitude Modulation (QAM) is one of the type of demodulator who can demodulate signal in high frequency.

In this thesis has ever been designed and implementated demodulator 64-QAM on FPGA (Field Programable Gate Array) by using programable language Very High Speed Integrated Cicuit (VHSIC) Hardware Description Language (VHDL). In this thesis has ever been implementated sub block from demodulator, such as demapper. The function of this block is to remapped input symbols with the different amplitude and phase who have been representated in to the information bits, but still in the inphase and quadrature bits.

The result of this thesis has been gotten output in the receiver for ideal condition is information bits which is same like an information bits in the transmitter side. When there is some errors, the output bits is same with input bits during the noise just intefere six bits from LSB (Least Significant bit, but for seven bits that are intefered there is 21,8310%, whereas fourteen bits that are interfered there is 96,9072%.

Keywords : 64-QAM, Digital Demodulation, FPGA, VHDL.

1. PENDAHULUAN

Pada bab ini dijelaskan tentang desain dan implementasi arsitektur *demapper* 64-QAM pada FPGA.

1.1 Latar Belakang

Sistem telekomunikasi bertujuan untuk mengirimkan sinyal dari sumber informasi ke tujuan yang diinginkan. Informasi yang akan dikirimkan akan diubah menjadi sinyal yang dapat dilewatimedia transmisi, dan agar sinyal yang diterima disisi penerima dapat dibaca, diperlukan *demodulator* yang dapat mengubah sinyal yang diterima menjadi informasi seperti yang dikirimkan. *Demodulator* 64-*QuadratureAmplitude*

Modulation (QAM) adalah salah satu jenis *demodulator*.

Demodulasi QAM berada pada sisi penerima, sistem kerjanya adalah merubah kembali bit-bit informasi yang telah diubah kedalam bentuk bit-bit simbol pada proses modulasi di sisi pengirim, menjadi bit-bit informasi yang sama dengan bit-bit informasi asal. Proses pemetaan balik dari simbol-simbol yang diterima dari pengirim menjadi bit-bit informasi dilakukan pada sub blok *demodulator*, yaitu pada *demapper*.

2. LANDASAN TEORI

Desain dan implementasi *demapper* 64-QAM dari sebuah sistem tertentu diperlukan adanya teori dasar yang mendukung.

2.1 Modulasi Digital

Pemilihan teknik modulasi sangat penting agar dapat mengatasi gangguan kanal komunikasi seperti *noise* ataupun interferensi. Umumnya pemilihan teknik modulasi, didasarkan pada kondisi kanal komunikasi, kebutuhan transfer data, efesiensi dan tingkat kerumitan rangkaian.

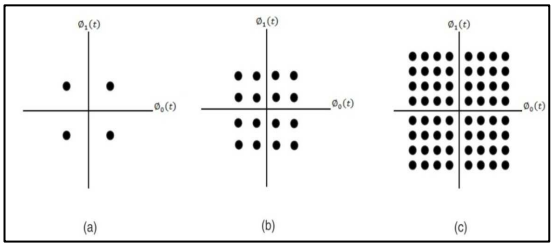
Pada modulasi digital sinyal informasi akan dikonversikan terlebih dahulu kedalam simbol digital untuk kemudian dilakukan proses modulasi. Modulasi digital merupakan proses penumpangan sinyal digital (*bit stream*) ke dalam sinyal pembawa (*carrier*). Modulasi digital sebetulnya adalah proses merubah karakteristik dan sifat gelombang pembawa sedemikian rupa sehingga bentuk hasilnya memiliki ciri-ciri dari bit-bit (0 atau 1) yang dikandungnya.

2.2 *Quadrature Amplitudo Modulation*

QAM merupakan suatu teknik modulasi digital yang mengkombinasikan modulasi amplitude dan fasa. Hal ini dapat dilakukan dengan memodifikasi gelombang pada kanal *inphase* dan *quadrature*. Sehingga gelombang yang menyusun konstelasinya selain memiliki fasa yang berbeda juga memiliki amplitudo yang bervariasi.

Pengubahan bit kesimbol berfungsi memetakan runtun bit informasi menjadi simbol QAM. Pada umumnya keluaran pengubah bit-ke-simbol akan dipetakan kebentuk kode Gray.

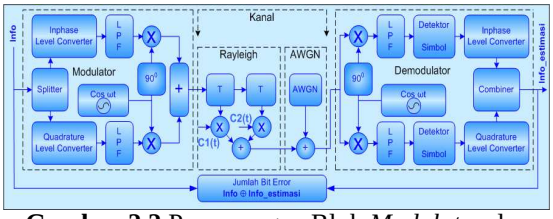
Orde QAM yang biasa digunakan adalah 4, 16, 64, ataupun 256. Konstelasi-konstelasi dari masing-masing orde QAM dapat dilihat seperti pada Gambar 2.1



Gambar 2.1 Konstelasi MQAM untuk (a) M = 4, (b) M = 16, (c) M = 64[7]

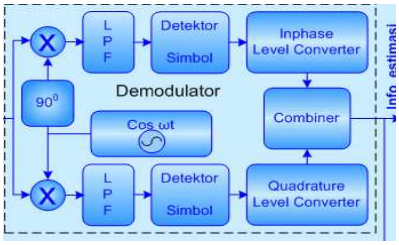
2.3 *Demodulator*

M-QAM tidak dapat terlepas dari sistem *modulator* yang berada di sisi pengirim dan *demodulator* yang berada disisi penerima.M-QAM pada sisi *modulator* berfungsi untuk menumpangkan sinyal informasi kedalam sinyal pembawa agar dapat dilewatkan kedalam kanal transmisi, sedangkan *demodulator* pada sisi penerima berfungi untuk mengembalikan sinyal informasi yang telah ditumpangkan ke sinyal pembawa menjadi sinyal informasi seperti sinyal pada input *modulator*.



Gambar 2.2 Perancangan Blok *Modulator* dan *Demodulator* Pada Matlab[11]

Sub blok pada *demodulator* terdiri dari *mixer*, *filter*, dan *demapper*. *Demapper* itu sendiri terdiri dari detektor simbol, *level converter* dan *combiner*. Detektor simbol berfungsi untuk mendeteksi posisi dari simbol-simbol yang diterimal. *Level converter* berfungsi untuk mengubah simbol-simbol yang direpresentasikan kedalam bit-bit menjadi bit-bit *inphase* dan *quadrature* seperti saat bit-bit dikirim disisi pengirim, sedangkan *combiner* berfungsi menggabungkan bit-bit *inphase* dan bit-bit *quadrature* atau dapat dikatakan perubahan bit-bit dari paralel ke serial.



Gambar 2.3 Blok *Demodulator* Pada Matlab[11]

3. PERANCANGAN SISTEM

Pada bab ini dijelaskan perancangan sistem *demapper* digital 64-QAM. Perancangan sistem meliputi penentuan spesifikasi sistem *demapper*, penentuan blok-blok sub sistem *demapper* dan pemodelan blok-blok sub sistem pada VHDL. Hasil dari pemodelan ini berupa *source- source* VHDL dapat disimulasikan pada Isim dan dapat diimplementasikan pada FPGA dengan *software* Xilinx ISE 14.4.

Permodelan tugas akhir ini mengacu pada Matlab[11] dan tugas akhir sebelumnya[2] serta didukung oleh teori[7]. Blok *Demapper* yang dibuat terdiri dari *Level Converter Inphase* dan *Quadrature*, serta *Combiner*.

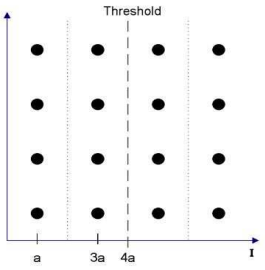
Tujuan tugas akhir ini adalah untuk menganalisis hasil *output* dari implementasi *demapper* digital 64-QAM. Hasil dari *Level converter* meghasilkan 3 bit *inphase* dan 3 bit *quadrature*, kemudian bit-bit tersebut digabungkan oleh *combiner*. Bentuk keluaran dalam pengerjaan tugas akhir ini adalah mendapatkan nilai bit-bit dari hasil implementasi pada FPGA yang dibandingkan dengan bit-bit yang dikirim oleh blok pengirim dari tugas akhir sebelumnya.

3.1 Parameter Perancangan Sistem *Demapper* 64-QAM

Parameter-parameter pada pemodelan sistem yang harus diperhatikan adalah algoritma penentuan nilai *threshold* serta jenis *noise* apa yang akan digunakan pada saat pengujian. Ada banyak algoritma dalam penentuan nilai *threshold*, diantaranya *Amplitude Estimation*, *Power Estimation*, dan *Improved Power Estimation*[6]. *Noise* yang digunakan berasal dari bit *generator*, dalam tugas akhir ini digunakan PRNG *(Pseudo Random Number Generator)*menggunakan LFSR (*Linier Feedback Shift Register*)[8].

3.1.1 Algoritma *Threshold*

Algoritma yang digunakan ntuk penentuan nilai *threshold* adalah algoritma *Improved Ampitude Estimation* (IAE)[6]. Algoritma ini melakukan estimasi langsung nilai *threshold* dari "one-dimensional axis amplitudes". Pada gambar 3.1 dapat dilihat batas *threshold* untuk sumbu *inphase*



Gambar 3.1 Nilai *threshold*disisi kanan atas sumbu *inphase* 64-QAM[6]

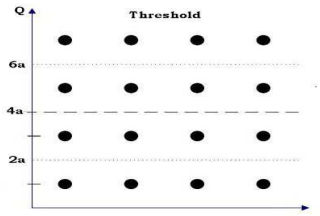
Pada gambar diatas sangat mudah menentukan nilai *threshold*, yaitu dari rata-rata posisi simbol.

$$Thr = \frac{a + 3a + 5a + 7a}{4} \quad (3.1)$$

dengan

$$a = \sqrt{\frac{P}{42}} \quad (3.2)$$

Menghitung nilai *threshold* cukup sekali, karena nilai *threshold* berikutnya dapat dihasilkan dengan mengkalikan atau membaginya. Karena konstelasi pada 64-QAM simetris, maka *Quadrant* kanan atas yang ditunjukkan pada gambar 3.1 berlaku untuk *Quadrant* lainnya. Gambar 3.2 menujukkan *threshold* sumbu *quadrature*.



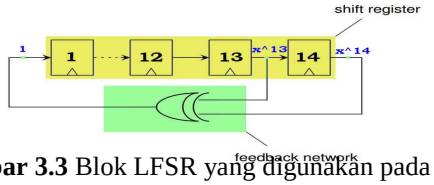
Gambar 3.2 Nilai *threshold*disisi kanan atas sumbu *quadrature* 64-QAM[6]

3.1.2 PRNG*(PseudoRandom Number Generator)*[8]

Pada tugas akhir ini fungsi *noise* diperankan oleh PRNG. PRNG ini pada dasarnya adalah bit yang dihasilkan seolah-olah *random*. PRNG yang digunakan adalah LFSR *(Linier Feedback Shift Register)*jenis Fibonacci.

LFSR yang digunakan adalah dengan polinomial:

$$P(x) = x^{14} + x^{13} + 1 \quad (3.3)$$

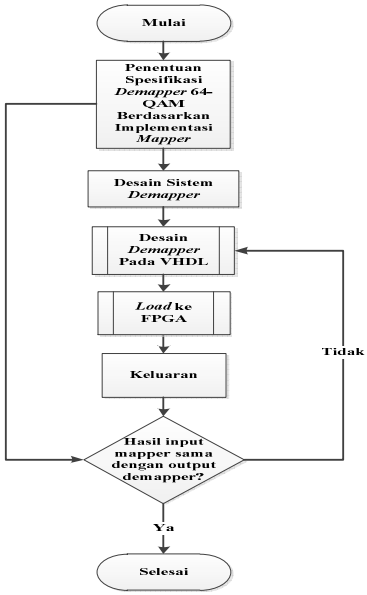


Gambar 3.3 Blok LFSR yang digunakan pada implementasi

3.2 Diagram Alir Perancangan Sistem *Demapper* 64-QAM

Pada sub bab ini dibahas tahapan-tahapan perancangan sistem *demapper* 64-QAM. Perancangan sistem diawali dengan pemilihan spesifikasi *demapper* 64-QAM, selanjutnya dibuat blok-blok sub sistem *demapper* 64-QAM spesifikasi tersebut. Kemudian blok-blok sub sistem tersebut dibuat pada VHDL.

Alur perancangan sistem *demapper* digital 64-QAM ditampilkan pada blok diagram pada gambar 3.7.



Gambar 3.5 Diagram Alir Perancangan *Demapper*

3.3 Penentuan Spesifikasi *Demapper* 64-QAM

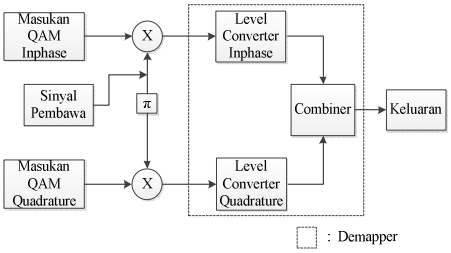
Penentuan parameter pada penelitian ini berdasarkan pada pemodelan berdasarkan teori pada bab sebelumnya dan pemodelan berdasarkan matlab. Dari kedua pemodelan tersbut ditentukan beberapa parameter yang digunakan pada sistem *demapper* yang akan dibuat pada penelitian ini.

3.3.1 Pemodelan Sistem Berdasarkan Teori

Pemodelan sistem berdasarkan teori dilakukan untuk mengetahui parameter-parameter sistem dan blok-blok sub sistem. Pemodelan ini berdasarkan pada teori yang telah dijelaskan pada Bab II.Spesifikasi sistem ini merupakan kumpulan parameter-parameter yang dijadikan dasar dalam perancangan sistem. Parameter-parameter didapat dari dasar teori yang telah dijelaskan pada Bab II.

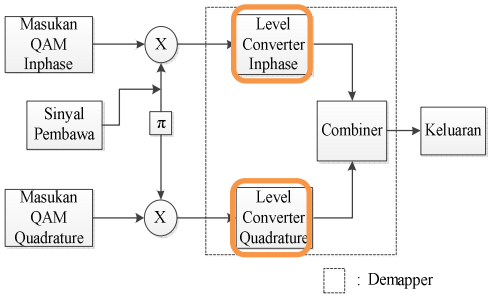
Berdasarkan parameter-parameter tersebut, pemodelan sistem de*modulator* digital 64-QAM digambarkan pada gambar 3.6.

Pada penelitian kali ini blok yang diimplementasikan hanya blok *demapper* saja. Hal ini dikarenakan sinyal pembawa merupakan sinyal analog, sedangkan fokus pada penelitian ini hanya pada sistem digitalnya saja.



Gambar 3.6 Blok Diagram De*modulator* 64-QAM

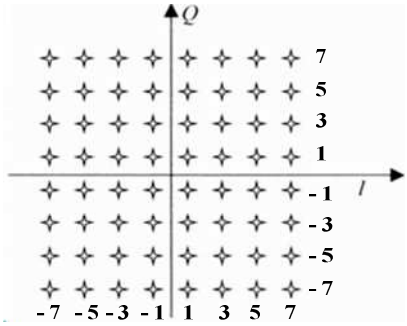
a. *Level Converter Inphase* dan *Quadrature*



Gambar 3.7*Level Converter* pada blok diagram *demodulator* 64-QAM

Level Converter Inphase dinotasikan sebagai bilangan real, sedangkan *Level Converter Quadrature* dinotasikan sebagai bilangan imajiner. Pembagian tersebut berdasarkan pada persamaan $N = \pm(\sqrt{M} - i)$, dimana nilai $i =$ bilangan ganjil dari 1 sampai dengan $\sqrt{M} - 1$. M menyatakan jumlah simbol yang dihasilkan, untuk 64-QAM nilai M = 64. 64 simbol tersebut menjadi delapan simbol di *Level ConverterInphase* dan delapan simbol di *Level ConverterQuadrature*. Delapan simbol masukan tersebut adalah -7, -5, -3, -1, 1, 3, 5, 7.

Delapan simbol masukan tersebut dikonstelasikan seperti gambar 3.8. Setiap simbol pada konstelasi tersebut dibentuk dari kombinasi delapan simbol di *Level ConverterInphase* dan delapan simbol di *Level ConverterQuadrature.*



Gambar 3.8 Konstelasi Simbol QAM[3]

Nilai keluaran dari simbol-simbol dari *Level Converter Inphase* dan *Level ConverterQuadrature* diperoleh dari persamaan energi rata-rata masing-masing simbol dengan persamaan sebagai berikut[5]

$$E_{avg} = \frac{2}{3}(M-1)A^2; \quad M = 64 \tag{3.4}$$

$$A = \sqrt{E_{avg}}\,\frac{1}{\sqrt{42}} \tag{3.5}$$

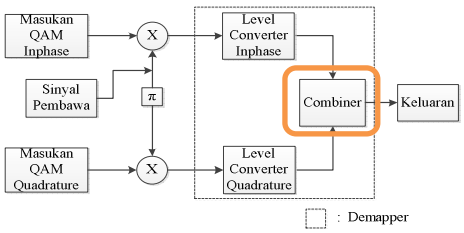
Nilai $\sqrt{E_{avg}}$ diasumsikan sebagai simbol masukan *Level ConverterInphase* dan *Level ConverterQuadrature*. Dengan demikian A merupakan nilai keluaran *Level ConverterInphase* dan *Level Converter Quadrature* dapat dinyatakan dalam tabel berikut

Tabel 3.1 Representasi level dan simbol *inphase* dan *quadrature*

Level Converter	Representasi Level	Representasi Simbol Keluaran
7	011	00001000101001
5	010	00000110001011
3	000	00000011101101
1	001	00000001001111
-1	101	11111110110001
-3	100	11111100010011
-5	110	11111001110101
-7	111	11110111010111

b. *Combiner*

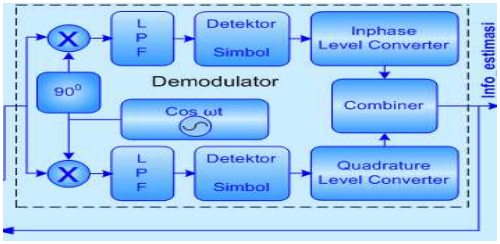
Pada blok ini dilakukan penggabungan output dari *level converter inphase* dan *quadrature* menjadi bentuk bit-bit informasi seperti bit-bit pada *input mapper*.



Gambar 3.9 *Combiner* pada Blok Diagram *Demodulator* 64-QAM

3.3.2 Pemodelan Sistem *Demapper* 64-QAM Berdasarkan Matlab[11]

Pemodelan *demapper* berdasarkan matlab merupakan hasil penelitian sebelumnya[11]. Pemodelan ini digunakan sebagai salah satu acuan dalam penentuan spesifikasi sistem *demapper*. Berikut pemodelan *demapper* berdasarkan blok *demodulator* dari hasil Matlab[11].



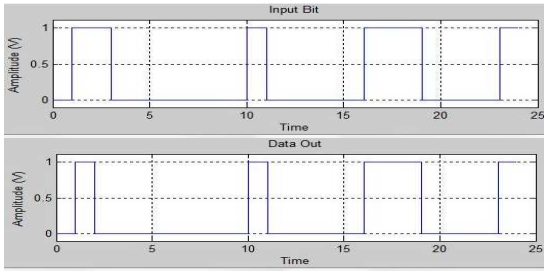
Gambar 3.10 *Demodulator* digital pada M-QAM Modem Demo pada Matlab

Berikut adalah perbandingan tampilan info *mapper* dengan *output combiner* pada pemodelan berdasarkan matlab dengan kondisi terdapat *error* satu bit, dan parameter-parameter dari pemodelan matlab dapat dilihat pada tabel 3.2.

Tabel 3.2 Parameter M-QAM Modem pada Matlab

Parameter	**Nilai**
Bit Rate	1,2 Kbps
Jenis Modulasi	64 QAM
Roll of Factor	0,00
Kanal	AWGN
SNR	25

a. **Info pada *mapper* vs *Combiner* pada *demapper***



Gambar 3.11 Perbandingan *Input* pada *Mapper* dengan *Output* pada *Combiner* dengan kondisi *error* satu bit

3.4 Representasi Bilangan

Dalam merancang sistem digital saat data masukan ataupun data hasil penghitungan dalam beberapa proses menghasilkan nilai dalam bentuk bilangan- bilangan desimal, sehingga perlu disepakati dari awal representasi bilangan dari bit-bit biner data yang diproses.

Pada tugas akhir ini dipilih format bilangan 2's *complement fixed-point* sepanjang 14 bit. Satu bit MSB (*Most Significant Bit*) untuk merepresentasikan *sign magnitude* sebagai tanda positif atau negatif, empat bit merepresentasikan bilangan desimal di depan koma, sedangkan sembilan bit LSB (*Least Significant Bit*) merepresentasikan bilangan di belakang koma.

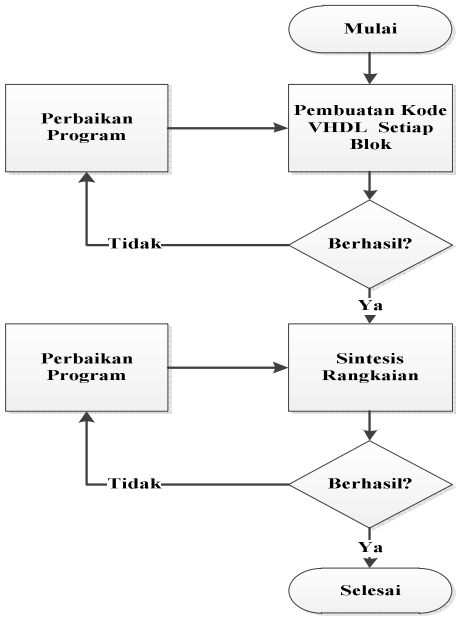
Representasi bilangan tersebut berdasarkan persamaan dibawah ini[5]

$$X = \frac{1}{512}\begin{cases} \sum_{n=0}^{N-1} x_n 2^n & ;X \geq 0 \\ 2^N - \sum_{n=0}^{N-1} x_n 2^n & ;X < 0 \end{cases} \quad (3.6)$$

dimana N adalah jumlah representasi bit, $x_n$adalah bit-bit biner ke-n berupa 0 atau 1 dan X merupakan representasi bilangan desimalnya dari 14 bit tersebut.

3.5 Desain *Demapper* Pada VHDL

Sebelum sistem *demapper* digital 64-QAM diimplementasikan pada *Field Programmable Gate Array* (FPGA), sistem harus dibuat pada VHDL. Model sistem *demapper* pada VHDL dibuat berdasarkan parameter *demapper* 64-QAM secara teori.

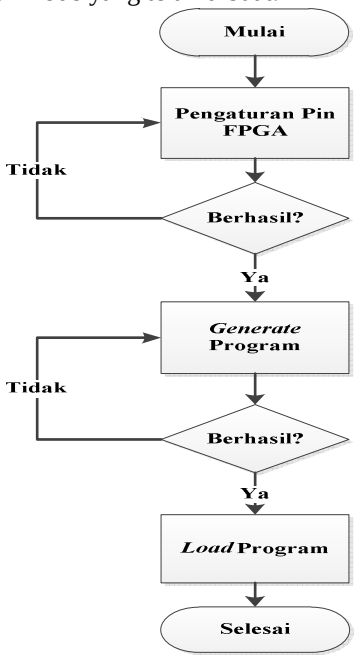


Gambar 3.12 Diagram Alir Desain Pada VHDL

3.6 *Load* ke FPGA

Proses *load* ke FPGA pada dasarnya adalah dengan mengatur peletakkan pin pada FPGA, kemudiaan melakukan proses *generate* program, lalu l*oad* program ke FPGA. Proses tersebut

dilakukan di *software* Xilinx 14.4. Pada proses *generate*, diubah program yang telah dibuat ke dalam format *.bit. format inilah yang akan ditanam ke FPGA. FPGA melakukan sesuai perintah dari kode yang telah dibuat.



Gambar 3.13 Diagram Alir *Load* ke FPGA

4. PENGUJIAN DAN IMPLEMENTASI SISTEM

Pada bab ini dibahas tentang pengujian dan implementasi system.

4.1 Implementasi *Demapper* Digital 64-QAM

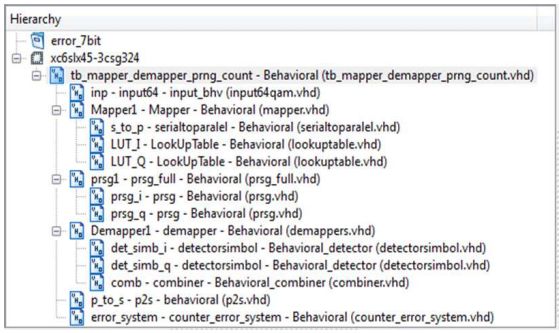
Implementasi dilakukan dengan menanamkan modul yang dirancang pada Xilinx ISE Project Navigator ke dalam FPGA menggunakan *software* Xilinx 14.4. Alur implementasinya adalah sebagai berikut

4.1.1Mendownloadkan bit file pada FPGA

Tahapan-tahapan yang dilakukan yaitu

a. Design Entry / Design utilities

Design Entry, yaitu menambahkan sumber kode VHDL yang telah dirancang ke dalam Xilinx 14.4.



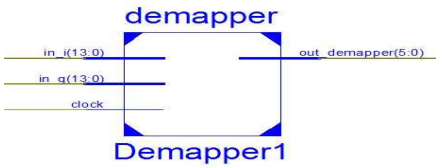
Gambar 4.1 *Design* Entry Rangkaian

b. User Constraints

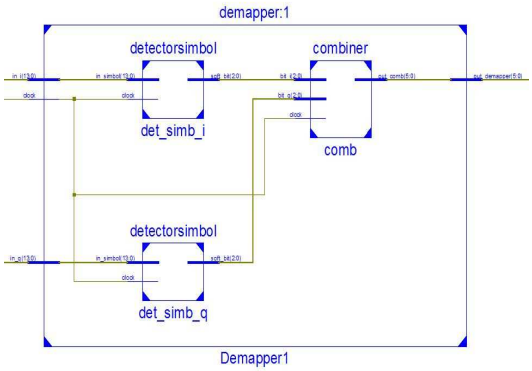
User Constraints, yaitu memetakan *port input/output* pada FPGA yang akan digunakan. Tahap ini adalah memetakan *port input/output* modul yang dirancang pada *port input/output* FPGA.

c. Synthesize

Synthesize, yaitu menerjemahkan kode VHDL ke dalam daftar jaringan (*netlist*) dan gerbang dasar pada FPGA



Gambar 4.2 Tampilan blok sistem *demapper* hasil sintesis



Gambar 4.3 Tampilan inti blok *demapper*

Dari laporan sintesis didapatkan keterangan hasil implementasi rangkaian sebagai berikut.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	32	54576	0%
Number of Slice LUTs	93	27288	0%
Number of fully used LUT-FF pairs	30	95	31%
Number of bonded IOBs	33	218	15%
Number of BUFG/BUFGCTRLs	1	16	6%

Gambar 4.4 Hasil Ringkasan Sintesis bagian *demapper*

d. Implement *Design*

Implement *Design*, dengan cara :

1. *Translate*, yaitu menerjemahkan netlist dan blok dasar berdasarkan *constraint*
2. *Map*, yaitu pemetaan rangkaian ke blok Xilinx FPGA, gerbang logika dan distribusinya.
3. *Place and Route*, yaitu penempatan hasil pemetaan pada blok dan gerbang yang dimaksud pada *floorplan*.

e. Generate Programming File

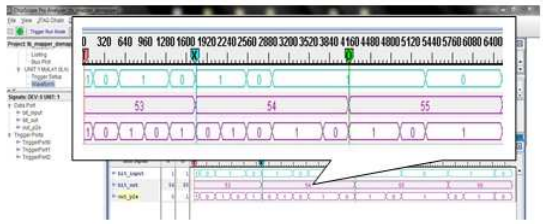
Generate Programming File, yaitu membuat file .bit yang akan dikirimkan pada FPGA melalui JTAG atau *downloader cable*.

4.4 Analisis Hasil Implementasi

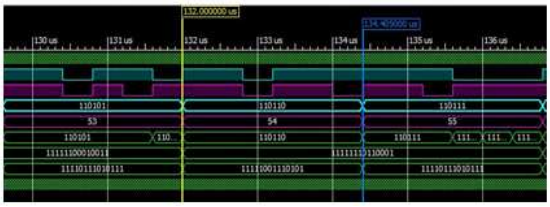
Analisis yang dilakukan adalah membandingkan hasil keluaran pada simulasi menggunakan Isim dengan keluaran pada *software* ChipScope. Skenario yang dilakukan adalah dengan kondisi ideal, kondisi diganggu 6 bit, kondisi di ganggu 7 bit, dan kondisi diganggu 14 bit.

4.4.1 Kondisi Ideal

(a)



(b)



Gambar 4.5 Perbandingan Hasil (a) ChipScope dengan (b) Isim M*apper-Demapper* Kondisi Ideal

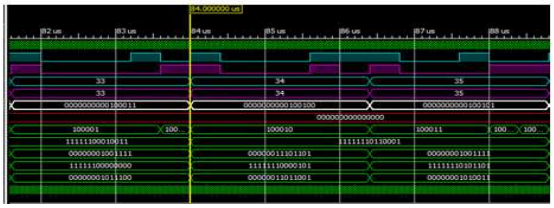
Pada gambar 4.5 (a) dapat dilihat bahwa *input serial* yang berwarna biru muda yang dimulai dari tanda X sampai tanda O bernilai 110111, sesuai dengan *output serial* maupun *prallel* yang berwarna ungu yang dimulai dari tanda O yang bernilai 110111. Jika dibandingkan dengan gambar 4.5 (b) , maka hasilnya sama, sehingga dapat dikatakan implementasi berhasil.

4.2.2 Kondisi Dengan *Noise*

a. Gangguan enam bit

(a)

(b)

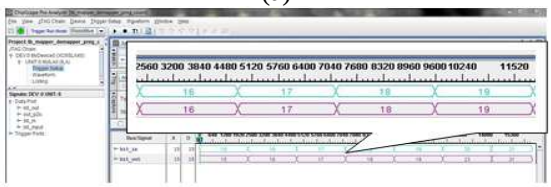


Gambar 4.6 Perbandingan Hasil (a) ChipScope dengan (b) Isim M*apper-Demapper* Kondisi Enam Bit Diganggu

Pada gambar 4.6 diatas dapat dilihat bahwa bit *input* yang ditandai warna biru, sama dengan bit *output* yang ditandai warna ungu. Pada pengujian ini bit *input* parallel disamakan *clock* nya dengan *output parallel*. Pada gambar 4.6 (a) dan (b) menghasilkan *output* yang sama, sehingga dapat dikatakan implementasinya berhasil.

b. Gangguan tujuh bit

(a)



(b)

Gambar 4.7 Perbandingan Hasil (a) ChipScope dengan (b) Isim M*apper-Demapper* Kondisi Tujuh Bit Diganggu

Pada gambar 4.7 diatas dapat dilihat bahwa bit *input* ditandai warna biru, sedangkan bit *output* ditandai warna ungu. Pada pengujian ini bit *input* parallel disamakan *clock* nya dengan *output parallel*. Pada gambar 4.7 (a) dan (b) menghasilkan bentuk bit yang sama. Dari segi *error*, pada simulasi Isim didapat *error process* sebesar 22,6389% dengan sample *process* 187.500 dan *error process* sebesar 42.448. Sedangakan pada Implementasi, diambil sample *process* 284 dengan *error process* 62 seperti pada tabel 4.1,

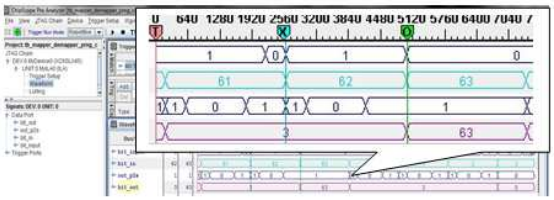
Tabel 4.1 *Sample Process Error* Tujuh Bit

Error tujuh bit	total
Sample Process	284
Error Process	62
Persentase	**21,83%**

Dari data tersebut dapat dikatakan implementasi berhasil.

c. Gangguan empat belas bit

(a)



(b)

Gambar 4.8 Perbandingan Hasil (a) ChipScope dengan (b) Isim M*apper-Demapper* Kondisi Empat Belas Bit Diganggu

Pada gambar 4.8 diatas dapat dilihat bahwa bit *input* ditandai warna biru, sedangkan bit *output* ditandai warna ungu. Pada gambar 4.8 (a) dan (b) menghasilkan bentuk bit hampir yang sama. Pada ChipScope, terlihat banyak sekali *process* yang *error*. Begitu pula pada Isim, namun pada Isim *output* terdapat lambang X, artinya *"unknown"*, karena *error*nya sudah melewati batas dari titik konstelasi yang di toleransi, dapat dilihat di gambar 3.4. Dari segi *error*, pada simulasi Isim didapat *error process* sebesar 99,2763% dengan sample *process* 187.500 dan *error process* sebesar 186.143. Sedangakan pada Implementasi, diambil sample *process* 291 dengan *error process* 282 seperti pada tabel 4.2,

Tabel 4.2 *Sample Process Error* empat belas Bit

Error empat belas bit	total
Sample Process	291
Error Process	282
Persentase	**96,91%**

dari data tersebut dapat dikatakan implementasi berhasil.

Setelah membandingkan hasil dari implementasi terhadap hasil simulasi, maka dapat disimpulkan bahwa *error* yang terjadi pada hasil implementasi memiliki toleransi perbedaan error dengan hasil simulasi kurang dari 5%.

5. PENUTUP

5.1. Kesimpulan

Dari hasil penelitian yang dilakukan, dapat disimpulkan bahwa :

a. Sistem yang didesain pada FPGA telah sesuai dengan sistem yang didesain pada matlab, maupun teori. Kesesuaiannya dapat dilihat dari gambar 4.2 dan 4.3.
b. *Demapper* digital 64-QAM dapat diimplementasikan pada FPGA. Dari hasil simulasi, sistem dapat melakukan pemetaan ulang hanya dengan delay 1 *clock* setelah *input* diterima lalu dikeluarkan *output* dalam *delay* 1 *clock* lagi.
c. Berdasarkan hasil sintesis blok sistem *demapper* 64-QAM didapatkan jumlah *resource* yang dibutuhkan adalah jumlah *slice registers* 32 atau 0% dari *resource* Spartan 6 XC6SLX45, jumlah *slice* LUTs 93 atau 0% dari *resource* Spartan 6 XC6SLX45, jumlah *fully used* LUT-FF *pairs* 30 atau 31% dari *resource* Spartan 6 XC6SLX45, dan jumlah *bonded* IOB 33 atau 15% dari *resource* Spartan 6 XC6SLX45. Dari hasil sintesis tersebut dapat disimpulkan bahwa hasil implementasi sistem tidak melebihi *source* pada FPGA sehingga dapat diimplementasikan.
d. Semakin banyak bit *input* yang diganggu, maka semakin besar *error process* yang terjadi, pada implementasi ini sistem *demapper* tahan jika bit yang diganggu tidak lebih dari enam bit dari LSB. Untuk tujuh bit yang diganggu dari LSB *error proces* yang terjadi adalah 21,8310%, sedangkan untuk empat belas bit yang diganggu maka *error process* yang terjadi adalah 96,9072%.

5.2. Saran

a. Lakukan penelitian dengan titik konstelasi yang lebih banyak, 128-QAM, 256-QAM dst.
b. Gunakan pemodelan kanal yang sebenarnya, misalnya seperti AWGN.
c. Agar lebih *real* perlu ditambah komponen *Analog to Digital Converter* (ADC).
d. Gunakan FPGA yang lebih rendah spesifikasinya agar tidak banyak *source* yang tidak digunakan.

DAFTAR PUSTAKA

[1] Ifeachor, Emmanuel C dan Barrie W Jervis. 2002. *Digital Signal Processing : A Practical Approach*. New Jersey : Prentice Hall

[2] Falaq, Fahrizal. 2013. Desain Dan Implementasi Modulator Digital 64-QAM Pada FPGA. Skripsi pada Institut Teknologi Telkom Bandung : Tidak diterbitkan.

[3] IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems.

[4] Nurkhalik, Subhan. 2011. Pengaruh Besar Orde QAM Pada OFDM Dengan Estimasi Kanal Mengunakan Interpolasi Linier. Skripsi pada Institut Teknologi Telkom Bandung : Tidak diterbitkan.

[5] Patmasari, Raditiana. 2012. Desain Arsitektur dan Implementasi Pengkode Konvolusi dan Pendekode Viterbi dengan Teknik Soft Decision pada Aplikasi DVB. Tesis pada Institut Teknologi Telkom Bandung : Tidak diterbitkan.

[6] Pitkänen, Sampo. 2008. Optimal reception of 64 Quadrature Amplitude Modulation in High-Speed Downlink Packet Access. Tesis Pada Helsinki University of Technology.

[7] Rice, Michael. 2008. *Digital Communications: A Discrete-Time Approach*. New Jersey : Pearson Prentice Hall.

[8] Schaumont, Patrick. 2008. A Random Number Generator in Verilog A Design Lecture.

[9] http://staff.ui.ac.id/internal/130781318/material/Chapter1.ppt (Akses tanggal 18 Maret 2012)

[10] http://www.digilentinc.com/Products/Detail.cfm?Prod=ATLYS (Akses tanggal 3 November 2012).

[11] http://www.mathworks.com/matlabcentral/fileexchange/15289-m-qam-modem-demo, (Akses tanggal 30 Maret 2013).