UNIVERSITAS INDONESIA

PERANCANGAN DAN IMPLEMENTASI *HUFFMAN CODING* UNTUK REDUKSI PAPR PADA SISTEM OFDM

SKRIPSI

BAMBANG BUDI HARTONO
0706199161

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2009

UNIVERSITAS INDONESIA

PERANCANGAN DAN IMPLEMENTASI *HUFFMAN CODING* UNTUK REDUKSI PAPR PADA SISTEM OFDM

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana teknik

BAMBANG BUDI HARTONO
0706199161

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	: Bambang Budi Hartono
NPM	: 0706199161
Tanda Tangan	: .....
Tanggal	: 29 Juni 2009

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Bambang Budi Hartono
NPM : 0706199161
Program Studi : Teknik Elektro
Judul Skripsi : Perancangan dan Implementasi *Huffman Coding* untuk Reduksi PAPR pada Sistem OFDM

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 : Prof. Dr. Ir. Harry Sudibyo DEA (……)

Penguji : Dr. Ir. Arman D. Diponegoro M.Eng (……..)

Penguji : Dr. Ir. Feri Yusivar M.Eng (……..)

Ditetapkan di : Depok, Universitas Indonesia
Tanggal : 29 Juni 2009

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Elektro pada Fakultas Teknik 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. Prof. Dr. Ir. Harry Sudibyo DEA, selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
2. Dr. Ir. Arman D. Diponegoro, M. Eng, yang telah banyak memberikan masukan dan saran;
3. Orang tua, yang telah memberikan bantuan dukungan material dan moral; dan
4. Sahabat, yang telah banyak membantu saya dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, 29 Juni 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Bambang Budi Hartono
NPM : 0706199161
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
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 *HUFFMAN CODING* UNTUK REDUKSI PAPR PADA SISTEM OFDM

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi 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 : 29 Juni 2009
Yang menyatakan

(Bambang Budi Hartono)

ABSTRAK

Nama : Bambang Budi Hartono
Program Studi : Teknik Elektro
Judul : Perancangan dan Implementasi *Huffman Coding* untuk Reduksi PAPR pada Sistem OFDM

Orthogonal Frequency Division Multiplexing atau OFDM merupakan teknik modulasi *multicarrier*, dimana antar *subcarriernya* satu dengan yang lain saling ortogonal. Karena sifat ortogonalitas ini, maka antar *subcarrier* yang berdekatan bisa dibuat *overlapping* tanpa menimbulkan efek *intercarrier interference* (ICI). Tetapi pada OFDM ini masih terdapat kelemahan dalam sistem transmisinya, yaitu masih tingginya *Peak to Average Power Ratio* (PAPR). *Peak to Average Power Ratio* atau PAPR merupakan perbandingan antara puncak daya maksimum terhadap harga daya rata-rata sinyal. Oleh karena itu, dibutuhkan suatu metode untuk mereduksi PAPR pada sistem OFDM.

Huffman Coding merupakan suatu metode kompresi data dengan cara pembentukan pohon *Huffman,* melalui proses *encoding* (pembentukan kode) menyebabkan data tersebut dapat dikompresi dan proses *decoding* (penguraian kode) sehingga data tersebut dapat diterjemahkan kembali menjadi kode aslinya. Teknik reduksi PAPR yang digunakan dalam skripsi ini adalah dengan menggunakan *Huffman Coding*. Metode *Huffman Coding* mampu untuk menurunkan PAPR sebesar 5 dB, sehingga *performance* OFDM semakin baik.

Kata Kunci : *Orthogonal Frequency Division Multiplexing* (OFDM), *Peak to Average Power Ratio* (PAPR), *Huffman Coding*, pohon *Huffman*, *encoding*, *decoding*.

ABSTRACT

Name : Bambang Budi Hartono
Study Program : Electrical Engineering
Title : Design and Implementation of Huffman Coding for the PAPR Reduction in OFDM System

Orthogonal Frequency Division Multiplexing or OFDM is a technique of multicarrier modulation, whose every subcarriers is orthogonal one another. Hence the nature of orthogonality, the nearby subcarrier can be overlapping without *intercarrier interference* (ICI) effect. But in the OFDM there are still some disadvantages in its transmition system, which is the over height of *Peak to Average Power Ratio* (PAPR). PAPR is the comparison of the top energy and the value energy in the mean signal. Therefore, there should be specific method to reduce PAPR in the OFDM system.

Huffman Coding is a data compression method by making the Huffman tree, through encoding process which is able to compress the data, and decoding process could be translated into the genuine code as well. The PAPR reduction technique which is discussed in this paper using Huffman Coding method. Huffman Coding method is able to reduce PAPR by 5 dB, so the OFDM performance would be good.

Keyword : Orthogonal Frequency Division Multiplexing (OFDM), Peak To Average Power Ratio (PAPR), Huffman Coding, Huffman tree, Encoding, Decoding.

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR SINGKATAN

OFDM : *Orthogonal Frequency Division Multiplexing*
FDM : *Frequency Division Multiplexing*
HPA : *High Power Amplifier*
BER : *Bit Error Rate*
AWGN : *Additive White Gaussian Noise*
ICI : *Intercarrier Interference*
PAPR : *Peak Average Power Ratio*
IFFT : *Invers Fast Tranformation Fourier*
FFT : *Fast Tranformation Fourier*
GI : *Guard Interval*
QAM : *Quadrature Amplitude Modulation*
IDFT : *Inverse Discrete Fourier Transform*
PSD : Power Spectral Density
BPSK : Binary Phase Shift Keying
QPSK : *Quadrature* Phase Shift Keying
PSK : Phase Shift Keying
ASK : *Amplitude* Shift Keying
AM : *Amplitude Modulation*
ISI : *Inter Symbol interference*

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia telekomunikasi sekarang ini berjalan sangat cepat, baik itu untuk hal hiburan, pendidikan, pemerintahan ataupun dunia usaha. Dengan berkembangnya hal tersebut maka kebutuhan adanya suatu sistem yang berguna untuk mengirimkan informasi dari satu atau banyak titik ke satu atau banyak titik lainnya semakin meningkat. Dalam proses pentransmisian data menggunakan frekuensi tertentu sebagai medianya, dibutuhkan suatu teknik tertentu yang mampu mengirimkan data ke tujuan yang diharapkan.

Teknik *Multiplexing* banyak digunakan untuk mengefisienkan proses pentransmisian data, salah satunya adalah OFDM. *Orthogonal Frequency Division Multiplexing* (OFDM) merupakan teknik modulasi *multicarrier,* dimana antar *subcarrier*-nya satu dengan yang lain saling ortogonal. Sifat ortogonalitas antar *subcarrier* yang berdekatan dapat dibuat *overlapping* tanpa menimbulkan efek *intercarrier interference* (ICI). Hal ini akan membuat sistem OFDM mempunyai efisiensi spektrum yang lebih tinggi jika dibandingkan dengan teknik modulasi *multicarrier* konvensional. Tetapi pada sistem OFDM masih terdapat kelemahan dalam sistem transmisinya yaitu masih tingginya *Peak Average Power Ratio* (*PAPR*). PAPR merupakan perbandingan antara puncak daya maksimum terhadap harga daya rata-rata sinyal. Tingginya PAPR menyebabkan kerugian antara lain *intermodulation* diantara *subcarrier,* besarnya penggunaan *bandwidth,* pemotongan puncak sinyal atau *Clipping.*

Dengan adanya kelemahan pada sistem OFDM akibat tingginya nilai PAPR maka perlu dikembangkan penelitian untuk mengatasi dan meningkatkan performansi sistem OFDM. Salah satu hal yang dapat dilakukan adalah melalui metode teknik *Huffman Coding. Huffman Coding* merupakan metode kompresi data yang dapat meminimumkan ukuran file. Diharapkan metode kompresi *Huffman Coding* dapat diimplementasikan pada sistem OFDM untuk menurunkan nilai PAPR (*Peak Average Power Ratio*).

1.2 Tujuan Penulisan

Tujuan penulisan skripsi ini adalah untuk merancang dan mengimplementasikan *Huffman Coding* untuk reduksi PAPR pada sistem OFDM menggunakan pemrograman matlab.

1.3 Ruang Lingkup

Skripsi ini melingkupi pembelajaran tentang sistem OFDM dengan kharakteristik *performance* dari PAPR, mempelajari sistem pemrosesan data per blok diagram dari OFDM, serta mempelajari konsep metode *Huffman Coding*. Kemudian menggabungkan semua sistem tersebut dan mensimulasikannya di pemrograman matlab

1.4 Pembatasan Masalah

Pembatasan masalah pada skripsi ini adalah sebagai berikut :

1. Pengkombinasian *Huffman Coding* pada sistem OFDM untuk menurunkan PAPR dengan parameter yang dianalisa adalah PSD (*Power Spectral Density*)
2. Tidak menggunakan HPA (*High Power Amplifier*) pada sistem transmisinya.
3. *Performance* BER (*Bit Error Rate*) dengan pengaruh SNR (*Signal Noise Ratio*) tidak diperhitungkan dalam skripsi ini.
4. *Channel* yang digunakan pada simulasi tidak dipengaruhi oleh *noise* AWGN.

1.5 Metode perancangan

Metode perancangan yang digunakan pada skripsi ini adalah studi literatur yang meliputi jurnal, artikel dan buku-buku tentang *Huffman Coding,* OFDM, teknik atau metode reduksi nilai PAPR pada sistem OFDM. Kemudian perancangan *Huffman Coding,* dan perancangan OFDM. Selanjutnya penggabungan sistem antara OFDM dengan *Huffman Coding* untuk menurunkan nilai PAPR.

1.6 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini terdiri dari bab-bab yang memuat beberapa sub-bab. Untuk memudahkan pembacaan dan pemahaman, maka laporan skripsi ini dibagi menjadi beberapa bab yaitu pada bab satu terdapat pendahuluan yang berisi tentang latar belakang, tujuan penulisan, ruang lingkup, pembatasan masalah, metode perancangan, dan sistematika penulisan. Pada bab 2 terdapat *Huffman Coding* dan sistem OFDM yang berisi tentang *Huffman Coding*, sistem OFDM, prinsip OFDM, modulasi digital, *Tranformation Fourier*, *Guard interval*, dan PAPR. Pada bab 3 terdapat perancangan model dan simulasi sistem yang berisi tentang pemodelan sistem, parameter simulasi, data *source*, *Huffman Coding*, *zero padding*, *serial to parallel* dan *parallel to serial*, modulasi dan demodulasi QAM, proses *Invers Fast Tranformation Fourier* (IFFT) dan (*Fast Tranformation Fourier*) FFT, *cycle prefix*. Pada bab 4 terdapat pengujian dan analisa yang berisi tentang pengujian dan analisa program *Huffman Coding* tanpa sistem OFDM dan pengujian dan analisa program *Huffman Coding* dengan sistem OFDM. Pada bab 5 terdapat kesimpulan dan saran dari skripsi ini.

BAB 2
HUFFMAN CODING DAN SISTEM OFDM

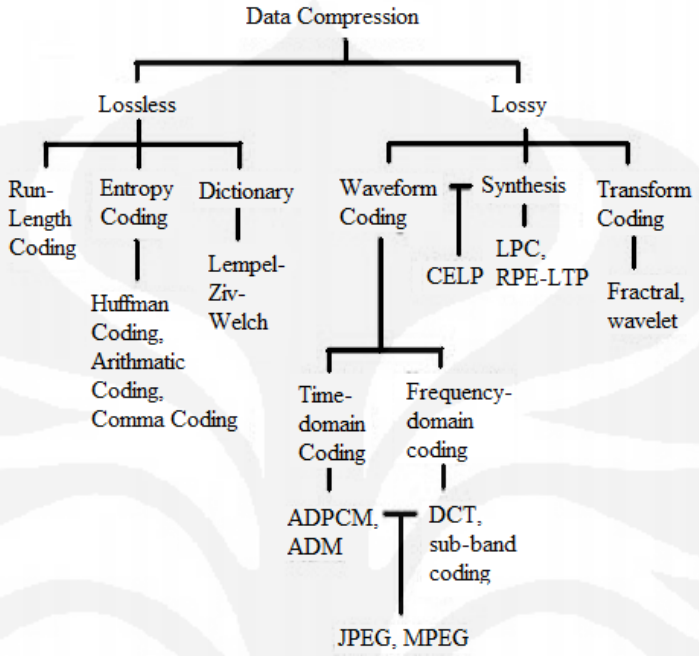
Kompresi data adalah hal yang esensial pada zaman modern ini. Hal yang penting dalam kompresi data adalah "rasio kompresi", atau rasio dari file yang dikompresi dengan file tersebut sebelum dikompresi. Misalnya, anggap sebuah data berukuran 100 kilobytes (KB). Dengan kompresi data dengan suatu *software* kompresi data, file tersebut katakanlah dapat direduksi menjadi 50 KB. Oleh karena itu, file tersebut lebih mudah untuk disimpan media penyimpan dan lebih mudah ditransmisikan melalui koneksi internet. Pada kasus spesifik ini, *software* kompresi tersebut dapat mereduksi ukuran data ini dengan factor pembagi 2, atau mereduksi data ini dengan "rasio kompresi" 2:1 [1].

Data kompresi terbagi menjadi kompresi data secara "*lossless*" dan "*lossy*". Kompresi data secara "*lossy*" adalah kompresi data yang hasil kompresinya, apabila didekompres, akan ada sebagian data yang hilang. Dengan kata lain, kompresi data secara "*lossy*" tidak dapat mendekompresi data seperti semula, sementara kompresi data secara "*loseless*" adalah kebalikannya. Kriteria pengkompresian :

1. Kualitas data hasil *encoding* : Ukuran hasil kompresi, tidak rusaknya data (*lossy*).
2. Ketepatan proses dekompresi data : Data hasil dekompresi dan sebelum kompresi tetap sama (*loseless*)
3. Kecepatan, rasio dan efisiensi proses kompresi dan dekompresi.

Kompresi data secara "*lossless*" ini digunakan ketika data yang didekompresi harus sama seperti aslinya (sebelum dikompresi). Data-data berbentuk tulisan (*text* files) misalnya, harus dikompresi menggunakan kompresi data secara "*lossless*", karena kehilangan sebuah karakter saja dapat berakibat pada kesalahpahaman pada kasus terburuk. Penyimpanan "*master source*" (sumber yang pasti/terpercaya) dari data gambar, video ataupun suara biasanya pun dikompresi secara "*loseless*". Akan tetapi, terdapat batasan yang tegas pada kemampuan mengompresi yang didapat dari kompresi data secara "*lossless*".

Rasio kompresi secara "*lossless*" biasanya berkisar antara 2:1 sampai 8:1 [1]. Berikut ini adalah gambar mengenai klasifikasi kompresi data yang ditunjukan pada Gambar 2.1.



Gambar 2.1 Klasifikasi Kompresi Data [1]

2.1 Sejarah *Huffman Coding*

Huffman Coding menggunakan tabel dengan variasi kode panjang untuk melakukan *encoding* dari sebuah simbol. Tabel variabel kode panjang tersebut telah dibuat terlebih dahulu secara terpisah berdasarkan nilai kekerapan munculnya suatu simbol. Metode ini ditemukan oleh David A. Huffman ketika ia melakukan studi Ph.D di MIT. Kode ini dipublikasikan pada tahun 1952 pada tulisannya yang berjudul "*A Method for the Constuction of Minimim-Redudancy Codes*". Biasanya *Huffman Coding* digunakan pada aplikasi seperti kompresi teks, data atau citra digital [4].

2.2 Pembentukan *Huffman Coding*

Huffman coding menggunakan metode spesifik untuk merepresentasikan setiap simbol yang menghasilkan suatu kode prefix. Kode prefix ini merupakan sekumpulan kode biner yang pada kode ini tidak mungkin terdapat kode prefix yang menjadi awalan bagi kode biner yang merepresentasikan simbol lain. Hal ini

akan mencegah timbulnya keraguan dalam proses *decoding.* Dalam *Huffman Coding*, kode biner untuk simbol dengan kekerapan lebih besar akan memiliki kode yang lebih pendek daripada untuk simbol dengan kekerapan lebih kecil [2].

Membentuk suatu *Huffman Coding* dimulai dengan membuat suatu pohon biner yang disebut pohon *Huffman.* Pohon ini akan disimpan pada suatu tabel, dengan ukuran yang bergantung pada jumlah dari simbol tersebut. Suatu simpul pada pohon biner dapat berupa simpul daun (simpul yang memiliki jumlah anak nol) ataupun simpul dalam (simpul yang mempunyai anak). Pada awalnya, semua simpul merupakan simpul daun, yang mengandung simbol itu sendiri serta bobotnya (frekuensi kekerapan) dari simbol tersebut dan bisa juga mengandung *link* ke simpul orangtua yang akan memudahkan pembacaan kode (secara terbalik) dimulai dari simpul daun. Pada simbul dalam terdapat bobot dan *link* ke dua simpul anak dan bisa ke simpul orangtua. Sebagai perjanjian, bit '0' akan merepresentasikan anak kiri dan bit '1' akan merepresentasikan anak kanan. Pohon yang telah selesai akan memiliki *n* buah simpul daun dan *n-1* buah simpul dalam [4].

Satu pohon *Huffman* dapat dibentuk dengan cara sebagai berikut [2] :

1. Membuat simpul daun sebanyak sejumlah simbol
2. Memilih dua simbol dengan peluang terkecil dan dikombinasikan sebagai suatu simpul orang tua.
3. Membuat simpul yang merupakan simpul orangtua dari dua simpul dengan peluang terkecil.
4. Memilih sebuah simpul berikutnya (termasuk simpul baru) yang memiliki peluang terkecil.
5. Melakukan prosedur yang sama pada dua simbol berikutnya yang memilki peluang terkecil.

2.3 Prinsip dasar algoritma *Huffman Coding* [10]

Prinsip dasar algoritma dalam *Huffman Coding* ini adalah bahwa setiap karakter misalnya ASCII biasanya diwakili oleh 8 bits. Jadi misalnya suatu file berisi deretan karakter "ABACAD" maka ukuran file tersebut adalah 6 x 8 bits = 48 bit atau 6 bytes. Jika setiap karakter tersebut di beri kode lain misalnya A=1,

B=00, C=010, dan D=011, berarti kita hanya perlu file dengan ukuran 11 bits (10010101011), yang perlu diperhatikan ialah bahwa kode-kode tersebut harus unik atau dengan kata lain suatu kode tidak dapat dibentuk dari kode-kode yang lain. Pada contoh diatas jika kode D kita ganti dengan 001, maka kode tersebut dapat dibentuk dari kode B ditambah dengan kode A yaitu 00 dan 1, tapi kode 011 tidak dapat dibentuk dari kode-kode yang lain. Selain itu karakter yang paling sering muncul, kodenya diusahakan lebih kecil jumlah bitnya dibandingkan dengan karakter yang jarang muncul. Pada contoh di atas karakter A lebih sering muncul (3 kali), jadi kodenya dibuat lebih kecil jumlah bitnya dibanding karakter lain. Untuk menentukan kode-kode dengan kriteria bahwa kode harus unik dan karakter yang sering muncul dibuat kecil jumlah bitnya, maka dapat menggunakan algoritma *Huffman*.

Sebagai contoh, sebuah file yang akan dimampatkan berisi karakter-karakter "PERKARA". Dalam kode ASCII masing-masing karakter dikodekan sebagai :

P = 50H = 01010000B

E = 45H = 01000101B

R = 52H = 01010010B

K = 4BH = 01001011B

A = 41H = 01000001B

Maka jika diubah dalam rangkaian bit, "PERKARA" menjadi :

01010000010001010101001001001011010000010101001001000001

P E R K A R A

yang berukuran 56 bit.

Kali pertama yang harus dilakukan adalah menghitung frekuensi kemunculan masing-masing karakter, jika kita hitung ternyata P muncul sebanyak 1 kali, E sebanyak 1 kali, R sebanyak 2 kali, K sebanyak 1 kali dan A sebanyak 2 kali. Jika disusun dari yang kecil :

E = 1

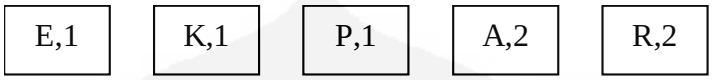
K = 1

P = 1

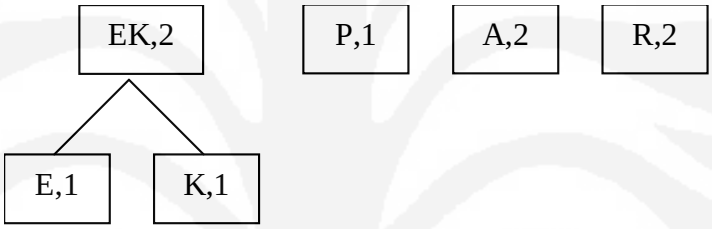
A = 2

R = 2

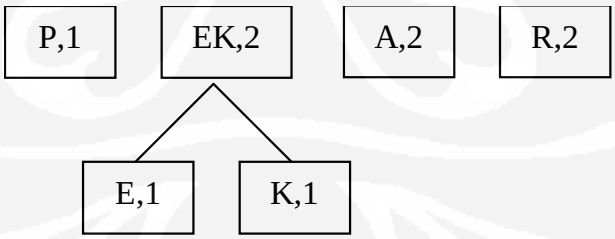
Untuk karakter yang memiliki frekuensi kemunculan sama seperti E, K dan P disusun menurut kode ASCII-nya, begitu pula untuk A dan R. Selanjutnya buatlah node masing-masing karakter beserta frekuensinya sebagai berikut :



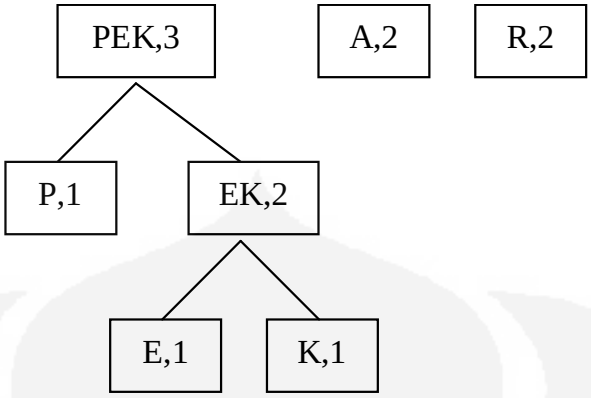
Ambil 2 node yang paling kiri (P dan E), lalu buat node baru yang merupakan gabungan dua node tersebut, node gabungan ini akan memiliki cabang masing-masing 2 node yang digabungkan tersebut. Frekuensi dari node gabungan ini adalah jumlah frekuensi cabang-cabangnya. Jika kita gambarkan akan menjadi seperti berikut ini :



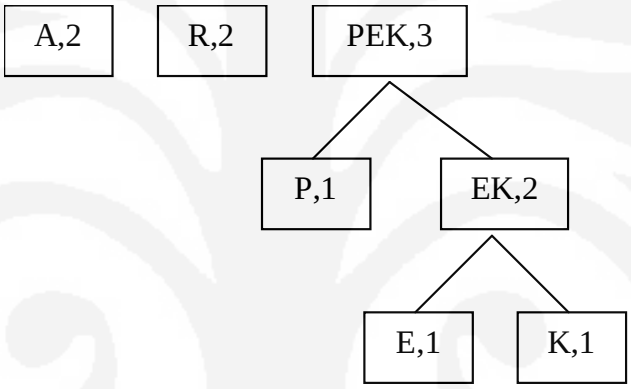
Jika kita lihat frekuensi tiap node pada level paling atas, EK=2, P=1, A=2, dan R=2. Node-node tersebut harus diurutkan lagi dari yang paling kecil, jadi node EK harus digeser ke sebelah kanan node P dan ingat jika menggeser suatu node yang memiliki cabang, maka seluruh cabangnya harus diikutkan juga. Setelah diurutkan hasilnya akan menjadi sebagai berikut :



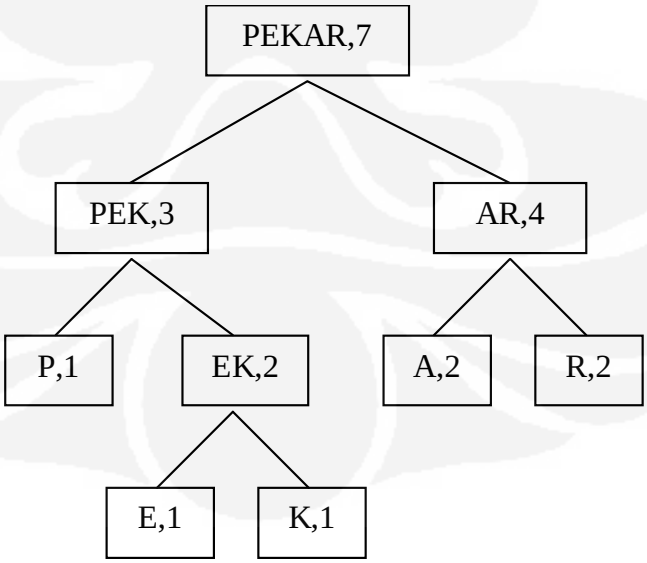
Setelah node pada level paling atas diurutkan (level berikutnya tidak perlu diurutkan), berikutnya kita gabungkan kembali 2 node paling kiri seperti yang pernah dikerjakan sebelumnya. Node P digabung dengan node EK menjadi node PEK dengan frekuensi 3 dan gambarnya akan menjadi seperti berikut ini :



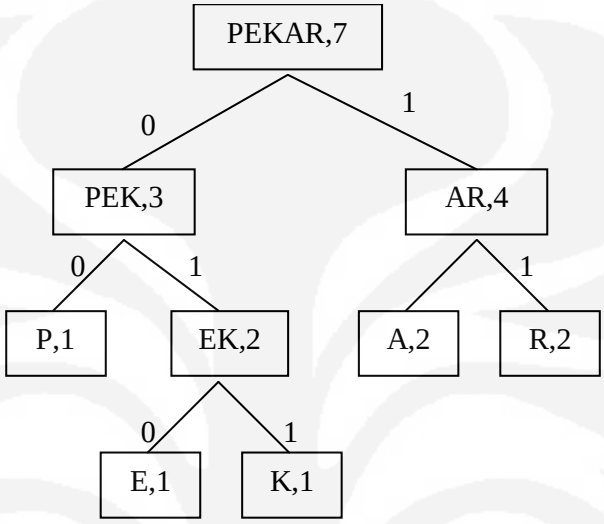
Kemudian diurutkan lagi menjadi :



Demikian seterusnya sampai diperoleh pohon *Huffman* seperti gambar berikut ini :



Setelah pohon *Huffman* terbentuk, berikan tanda bit 0 untuk setiap cabang ke kiri dan bit 1 untuk setiap cabang ke kanan. Berikut ini adalah hasil pembentukan pohon *Huffman* untuk string "PERKARA" yang ditunjukan pada Gambar 2.2.



Gambar 2.2 Pohon *Huffman* untuk String "PERKARA"

Untuk mendapatkan kode *Huffman* masing-masing karakter, telusuri karakter tersebut dari node yang paling atas (PEKAR) sampai ke node karakter tersebut dan susunlah bit-bit yang dilaluinya.

Untuk mendapatkan kode Karakter E, dari node PEKAR kita harus menuju ke node PEK melalui bit 0 dan selanjutnya menuju ke node EK melalui bit 1, dilanjutkan ke node E melalui bit 0, jadi kode dari karakter E adalah 010.

Untuk mendapatkan kode Karakter K, dari node PEKAR kita harus menuju ke node PEK melalui bit 0 dan selanjutnya menuju ke node EK melalui bit 1, dilanjutkan ke node K melalui bit 1, jadi kode dari karakter K adalah 011.

Untuk mendapatkan kode Karakter P, dari node PEKAR kita harus menuju ke node PEK melalui bit 0 dan selanjutnya menuju ke node P melalui bit 0, jadi kode dari karakter P adalah 00.

Untuk mendapatkan kode Karakter A, dari node PEKAR kita harus menuju ke node AR melalui bit 1 dan selanjutnya menuju ke node A melalui bit 0, jadi kode dari karakter A adalah 10.

Terakhir, untuk mendapatkan kode Karakter R, dari node PEKAR kita harus menuju ke node AR melalui bit 1 dan selanjutnya menuju ke node R melalui bit 1, jadi kode dari karakter R adalah 11. Hasil akhir kode *Huffman* dari file di atas adalah :

E = 010

K = 011

P = 00

A = 10

R = 11

Untuk proses pengembalian ke file aslinya, kita harus mengacu kembali kepada kode *Huffman* yang telah dihasilkan, seperti contoh di atas hasil pemampatan adalah :

000101101100 1110

dengan Kode *Huffman* :

E = 010

K = 011

P = 00

A = 10

R = 11

Ambillah satu-persatu bit hasil pemampatan mulai dari kiri, jika bit tersebut termasuk dalam daftar kode, lakukan pengembalian, jika tidak ambil kembali bit selanjutnya dan jumlahkan bit tersebut. Bit pertama dari hasil pemampatan di atas adalah 0, karena 0 tidak termasuk dalam daftar kode kita ambil lagi bit kedua yaitu 0, lalu digabungkan menjadi 00, jika kita lihat daftar kode 00 adalah kode dari karakter P.

Selanjutnya bit ketiga diambil yaitu 0, karena 0 tidak terdapat dalam daftar kode, kita ambil lagi bit keempat yaitu 1 dan kita gabungkan menjadi 01. 01 juga tidak terdapat dalam daftar, jadi kita ambil kembali bit selanjutnya yaitu 0 dan digabungkan menjadi 010. 010 terdapat dalam daftar kode yaitu karakter E. Demikian selanjutnya dikerjakan sampai bit terakhir sehingga akan didapatkan hasil pengembalian yaitu PERKARA.

2.4 Proses *Encoding*

Proses untuk melakukan pembentukan kode dari suatu data tertentu disebut *encoding*. Dalam hal ini, kode *Huffman* akan terbentuk sebagai suatu kode biner. Kode *Huffman* didapatkan dengan membaca setiap kode dari daun simbol tersebut hingga keakarnya. Ketika suatu kode *Huffman* telah dibentuk, suatu data dapat akan mudah di *encode* dengan mengganti setiap simbol menggunakan kode yang telah dibentuk [2].

Sebagai contohnya adalah pada Gambar 2.2 dapat dilihat telah terbentuk pohon *Huffman* dari string "PERKARA" dan perbandingan hasil kode *ASCII* dan kode baru yang dibentuk oleh *Huffman Coding* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan Kode ASCII dan Kode *Huffman* dari String "PERKARA"

Huruf	Kode Ascii	Kode *Huffman*
E	01000101	010
K	01001011	011
P	01010000	00
A	01000001	10
R	01010010	11

Pada saat karakter-karakter "PERKARA" akan diproses, maka data yang akan dikirim menjadi:

00 010 11 011 10 11 10 = 16 bit

P E R K A R A

Dengan Algoritma *Huffman* berarti file ini dapat kita hemat sebanyak 56-16 = 40 bit.

2.5 Proses *Decoding*

Penguraian kode (*Decoding*) adalah sebuah proses untuk menyusun kembali data yang telah dikodekan sebelumnya sehingga informasi yang diterima dapat dibaca dan diolah. Penguraian kode (*Decoding*) ini adalah lawan dari pengkodean (*Encoding*). Ada 2 cara penguraian kode *Huffman*. Cara yang pertama adalah

menggunakan pohon *Huffman* sedangkan cara yang kedua adalah menggunakan tabel kode *Huffman* [5].

Berikut adalah penjelasan untuk cara pertama, yaitu menggunakan pohon *Huffman.* Seperti yang dijelaskan sebelumnya, pohon *Huffman* adalah pohon biner dengan menggunakan kode awalan (*prefix code*). Hal ini memudahkan proses penguraian kode. Langkah-langkah yang dilakukan dalam penguraian kode (*decoding*) menggunakan pohon *Huffman* adalah sebagai berikut:

1. Baca bit pertama dari string biner masukan.
2. Lakukan traversal pada pohon *Huffman* mulai dari akar sesuai dengan bit yang dibaca. Jika bit yang dibaca adalah 0 maka baca anak kiri, tetapi jika bit yang dibaca adalah 1 maka baca anak kanan.
3. Jika anak dari pohon bukan daun (simpul tanpa anak) maka baca bit berikutnya dari string biner masukan.
4. Hal ini diulang (traversal) hingga ditemukan daun.
5. Pada daun tersebut simbol ditemukan dan proses penguraian kode selesai.
6. Proses penguraian kode ini dilakukan hingga keseluruhan string biner masukan diproses.

Contoh cara penguraian kode menggunakan pohon *Huffman.* Dengan menggunakan kode hasil enkripsi yang telah ditunjukkan dalam proses pengkodean (*Encoding*) sebelumnya, akan ditunjukkan cara penguraian kode (*decoding*) menggunakan pohon *Huffman*. Hasil pengkodean string “PERKARA” ke dalam string biner adalah 00 010 11 011 10 11 10. Bit pertama dari string biner tersebut adalah 0, dengan menggunakan pohon *Huffman* yang ditunjukkan dalam Gambar 2.2, ditemukan bahwa anak kiri nya bukanlah sebuah daun. Oleh karena itu harus diambil bit berikutnya yaitu bit kedua (angka 0). Karena bit kedua adalah 0 maka harus diambil anak kiri. Pada anak kiri tersebut ditemukan sebuah daun yang menyimpan simbol P. Dengan melakukan hal ini secara berulang hingga bit terakhir, maka akan ditemukan bahwa string biner 00 010 11 011 10 11 10 adalah hasil pengkodean (*enkripsi*) dari string “PERKARA”.

Cara kedua untuk menguraikan kode *Huffman* adalah dengan menggunakan tabel kode *Huffman*. Oleh karena kode *Huffman* disusun menggunakan kode awalah (*prefix code*) maka dapat dipastikan bahwa sebuah kode untuk sebuah

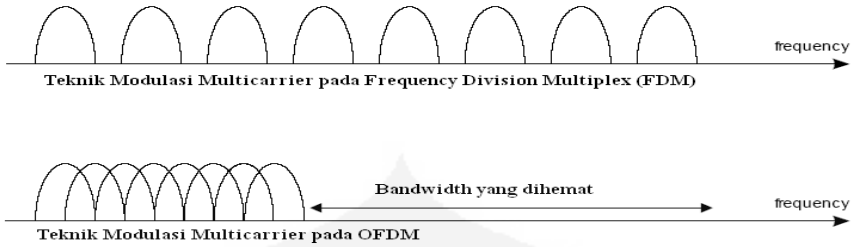
simbol/karakter yang satu tidak boleh menjadi awalan dari kode/simbol yang lain. Oleh karena itu pastilah string biner yang berisi hasil enkripsi dapat dipisahkan dengan mudah berdasarkan setiap rangkaian bitnya untuk diuraikan menjadi informasi semula. Yang perlu dilakukan hanyalah melihat setiap rangkaian bit yang ditemukan dalam string biner hasil enkripsi di dalam tabel kode *Huffman*.

Berikut ini akan diberikan contoh cara penguraian kode dengan menggunakan tabel kode *Huffman*. Misalkan akan dilakukan penguraian kode (*decoding*) berdasarkan string biner yang dihasilkan dari proses pengkodean (*encoding*) sebelumnya. String biner yang dihasilkan sebelumnya adalah 00010110111011 dengan tabel kode *Huffman* yang ditunjukkan pada Gambar 2.2. String biner tersebut dapat dipisahkan menjadi rangkaian bit : 00 010 11 011 10 11. Hal ini dapat dilakukan dengan mudah karena penyusunan rangkaian bit tersebut menggunakan kode awalan (*prefix code*). Setelah string biner tersebut dipisah menjadi rangkaian bitnya, hanya perlu dilihat rangkaian bit yang bersesuaian dengan tabel kode *Huffman* yang ditunjukkan pada Tabel 2.1. Maka akan dihasilkan string hasil penguraian kode (*dekripsi*), yaitu “PERKARA”.

2.6 Sistem OFDM

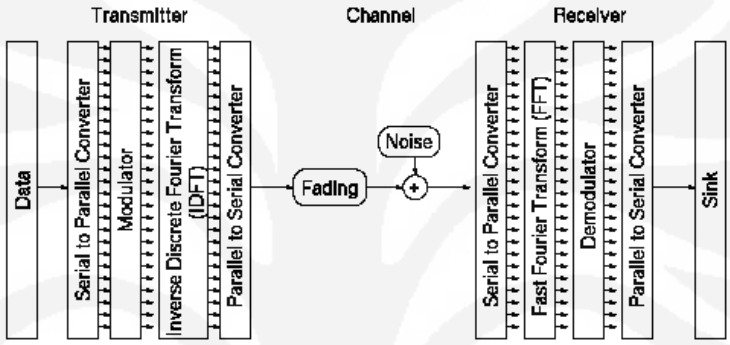
OFDM (*Orthogonal Frequency Division Multiplexing*) adalah sebuah teknik transmisi yang menggunakan beberapa buah frekuensi (*multicarrier*) yang saling tegak lurus (*orthogonal*) [8]. Masing-masing sub-carrier tersebut dimodulasikan dengan teknik modulasi konvensional pada rasio simbol yang rendah.

Teknologi OFDM hampir sama dengan *Frequency Division Multiplexing* (FDM) dalam akses tiap pengguna yaitu membagi *bandwidth* yang ada menjadi beberapa kanal yang dialokasikan ke tiap user, hanya saja pada OFDM menggunakan spektrum yang lebih efesien dengan spasi antar pengguna yang lebih dekat. Ini bisa dilakukan dengan membuat semua *subcriber* saling *orthogonal*, hal ini dikarenakan untuk menghindari interferensi antar pengguna yang berdekatan. Gambar 2.3 menunjukan perbandingan antara teknik OFDM dan FDM dengan sifat ortogonalitasnya.



Gambar 2.3 Perbedaan Teknik FDM dan OFDM

Berikut ini adalah gambar diagram blok sistem OFDM ditunjukan pada Gambar 2.4.



Gambar 2.4 Diagram Blok Sistem OFDM Secara Umum [8]

Prinsip kerja dari OFDM dapat dijelaskan sebagai berikut. Deretan data informasi yang akan dikirim dikonversikan kedalam bentuk parallel, sehingga bila bit rate semula adalah *R* , maka bit rate di tiap-tiap jalur parallel adalah *R/M* dimana *M* adalah jumlah jalur parallel (sama dengan jumlah sub-carrier). Setelah itu, modulasi dilakukan pada tiap-tiap *subcarrier*. Modulasi ini bisa berupa BPSK, QPSK, QAM atau yang lain, tapi ketiga teknik tersebut sering digunakan pada OFDM. Kemudian sinyal yang telah termodulasi tersebut diaplikasikan ke dalam *Inverse Discrete Fourier Transform* (IDFT), untuk pembuatan simbol OFDM. Penggunaan IDFT ini memungkinkan pengalokasian frekuensi yang saling tegak lurus (*orthogonal)*. Setelah itu simbol-simbol OFDM dikonversikan lagi kedalam bentuk serial, dan kemudian sinyal dikirim. [3][8].

Sinyal *carrier* dari OFDM merupakan penjumlahan dari banyaknya *subcarriers* yang *orthogonal*, dengan data *baseband* pada masing-masing *subcarriers* dimodulasikan secara bebas menggunakan teknik modulasi QAM atau PSK. [3]

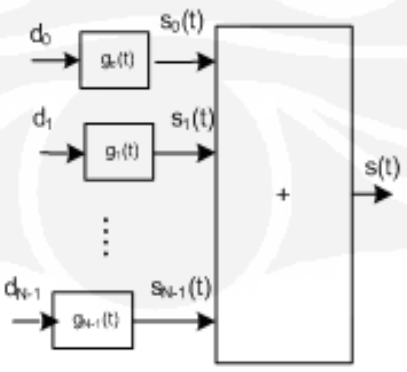
Sinyal yang terkirim tersebut, dalam persamaan matematik bisa diekspresikan pada persamaan 2.1

$$s(t) = \mathrm{Re}\{\sum_{n=-\infty}^{+\infty} b_n f(t-nT) e^{j(\omega_0 t+\varphi)}\}_{(1)} \quad (2.1)$$

Dengan :

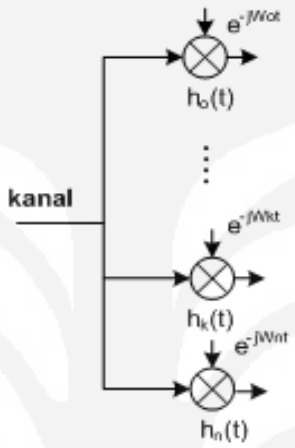
Re(.) adalah bagian real dari persamaan, *f(t)* adalah respons impuls dari filter transmisi, *T* adalah periode simbol, v_o adalah frekuensi pembawa (*carrier frequency*) dalam bentuk radian, j adalah fase pembawa (*carrier phase*), dan b_n adalah data informasi yang telah termodulasi yang menjadi input dari IDFT. [8].

Proses pentransmisian sinyal OFDM pada intinya menggunakan proses IFFT (*Invers Fast Fourier Transform*) untuk menghasilkan simbol–simbol OFDM yang saling *orthogonal* satu sama lain sehingga bila diamati dalam domain frekuensi, spektrum simbol-simbol tersebut saling *overlapping* tetapi tidak *interference.* Proses awal dari pemancar OFDM adalah aliran data dengan laju besar dibagi terlebih dahulu menjadi beberapa *subcarrier* sehingga laju data menjadi lebih kecil dari laju data awal. Dengan adanya pembagian aliran data yang besar menjadi aliran data dengan laju yang lebih kecil maka bila terjadi gangguan pada salah satu *subcarrier* maka hal tersebut tidak berpengaruh pada *subcarrier* yang lain sedangkan bila kita hanya menggunakan *single carrier* tidak OFDM (*multicarrier*) maka bila terjadi *interference* maka *interference* tersebut akan mempengaruhi keseluruhan data terkirim. Hal ini jelas sangat merugikan dalam dunia transmisi data khususnya di bidang telekomunikasi. Konsep pemancar OFDM ditunjukkan pada Gambar 2.5.



Gambar 2.5 Konsep Pemancar OFDM

Pada stasiun penerima, dilakukan operasi yang berkebalikan dengan apa yang dilakukan di stasiun pengirim. Mulai dari konversi dari serial ke parallel, kemudian konversi sinyal parallel dengan *Fast Fourier Transform* (FFT), setelah itu demodulasi, konversi parallel ke serial, dan akhirnya kembali menjadi bentuk data informasi. Konsep *multicarrier* pada penerima dapat ditunjukan pada Gambar 2.6



Gambar 2.6 Konsep Teknik *Multicarrier* pada Penerima

Pada penerima OFDM pada intinya menggunakan *Fast Fourier Transform* (FFT). FFT ini berfungsi untuk memisahkan sinyal yang diterima dari *carrier*nya. Hal tersebut dapat dilakukan dengan mengalikan sinyal yang diterima dengan *conjugate carrier*nya. Jadi kalau pada pemancar OFDM, sinyal pada tiap *subcarrier* yaitu d_n akan dikalikan dengan *subcarrier* masing – masing sebesar $e^{j\omega_n t}$ maka pada penerima OFDM, sinyal yang diterima oleh *receiver* terlebih dahulu akan diubah ke dalam beberapa *subcarrier* dan sinyal pada masing – masing *subcarrier* akan dikalikan dengan *conjugate subcarrier* pada pemancar OFDM yaitu $e^{-j\omega_n t}$ dengan $0 \le t < T_s$.

2.7 Prinsip OFDM

Pada OFDM, frekuensi-frekuensi *multicarrier* tersebut saling tegak lurus, yang berarti bahwa *crosstalk* di antara *sub-channels* dihilangkan dan *inter-carrier guard bands* tidak diperlukan. Istilah *orthogonal* dalam *Orthogonal Frequency Division Multiplexing* (OFDM) mengandung makna hubungan matematis antara

frekuensi-frekuensi yang digunakan. Ortogonal merupakan sifat matematika dari dua vektor yang saling tegak lurus. Pada sistem komunikasi, sinyal-sinyal dikatakan ortogonal jika mereka berdiri sendiri tanpa saling mengganggu satu sama lain. Sifat ortogonal dari vektor sinyal ini memungkinkan beberapa sinyal informasi dikirimkan pada kanal yang sama tanpa mengalami interferensi. Dua set sinyal dikatakan ortogonal, jika integral perkalian keduanya dalam satu interval sama dengan nol. Kondisi orthogonal ditunjukkan dengan formula

$$\int_a^b \varphi_p(t)\varphi_q^*(t)dt = 0 \qquad (2.2)$$

Untuk $p \neq q$

Satu prinsip kunci dari OFDM adalah skema modulasinya dengan rasio simbol yang rendah, sehingga hanya mendapat sedikit pengaruh *intersymbol interference* dari *multipath fading*. Oleh karena itu, maka dapat ditransmisikan sejumlah aliran *low-rate* dalam paralel, bukan aliran *high-rate* tunggal. Karena durasi dari tiap simbol panjang, maka memungkinkan untuk penyisipan *guard interval* di antara simbol-simbol OFDM, sehingga dapat menghilangkan *intersymbol interference* [3].

Kecepatan bit informasi yang dikirim sistem OFDM yang tinggi dibagi dan ditransmisikan pada sejumlah laju rate yang rendah. Bit informasi yang dikirimkan dibagi dalam subcarrier yang saling *orthogonal*, di mana masing-masing *carrier* tersebut saling *overlapping*, namun tidak saling menginterferensi antar *carrier* yang berdekatan. Masing-masing *carrier* dikatakan *linier independen* jika spasi masing-masing *carrier* adalah kelipatan $\frac{1}{T}$ [7].

Spektrum frekuensi kanal pada OFDM dapat ditumpangtindihkan dan tidak terjadi saling interferensi antar kanal, sebab *null* dari setiap kanal yang berdekatan jatuh tepat pada titik tengah spektrum yang membawa informasi (spektrum yang memiliki *power* tertinggi). Untuk mengatur supaya setiap *null* dari kanal spektrum tetangga jatuh tepat pada titik tengah spektrum yang membawa informasi, setiap sinyal transmisi pada setiap kanal harus bersifat saling *orthogonal* dan saling *harmonic*. Secara matematis, untuk membuat setiap sinyal *orthogonal* adalah

dengan membuat luas area positif sama dengan luas area negatif atau hasil integral dari sinyal tersebut adalah nol. Selanjutnya untuk *harmonic,* misalkan *c* adalah frekuensi pembawa dalam suatu *bandwidth* dengan persamaan $c_n = n \times c_1$, maka frekuensi c_n dikatakan *harmonic* dengan c_1, jika *n* adalah integer. Jika mereka juga saling *orthogonal*, maka ketika digabungkan, mereka tidak saling menginterferensi [6].

Pada proses pengiriman OFDM sinyal informasi dikirim melalui masing-masing subkanal, di mana banyaknya subkanal sama dengan banyaknya *subcarrier* yaitu sebanyak N. Akibatnya terjadi penurunan *bitrate* sebesar faktor N. Hal ini juga menyebabkan periode simbol meningkat N kali semula. Pada domain frekuensi, *bandwidth* menjadi N kali lebih kecil daripada *bandwidth* sinyal. Sinyal pada masing-masing *sub-channel* dimodulasi pada frekuensi tertentu, kemudian sinyal termodulasi tersebut dijumlahkan terlebih dahulu kemudian dikirimkan. Frekuensi masing-masing *subcarrier* harus tetap terjaga ke-*orthogonal*-annya agar tidak saling berinterferensi. Pada proses simulasi dengan Matlab, proses tersebut cukup dapat diimplementasikan dengan suatu transformasi *Fourier*.

2.8 Modulasi Digital

Modulasi digital dalam skripsi ini berfungsi untuk merubah data berupa bit-bit atau integer menjadi bilangan kompleks yang terdiri dari bilangan real dan imajiner dengan menggunakan *mapper*. Modulasi yang digunakan dalam skripsi ini adalah modulasi QAM (*Quadrature Amplitude Modulation*).

2.8.1 Modulasi QAM

QAM merupakan kombinasi modulasi antara ASK dan PSK. Pada QAM *fase* dan *amplitude* dari sinyal carrier diubah-ubah untuk melambangkan data. Persamaan dasar sinyal QAM dapat dituliskan sebagai berikut:

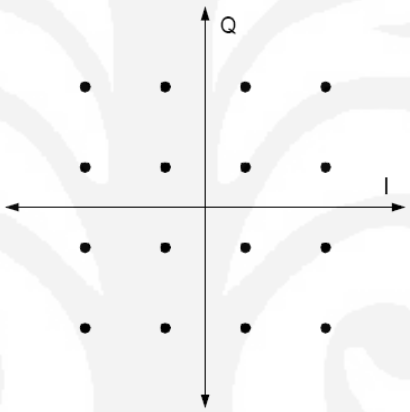
$$s(t) = I(t).\cos\omega_c t + Q(t).-\sin\omega_c t \qquad (2.3)$$

Dengan:

$$\begin{aligned} I(t) &= A.\cos\theta \\ Q(t) &= A.\sin\theta \end{aligned} \qquad (2.4)$$

Dari persamaan 2.3 dan 2.4, dapat dilihat bahwa sinyal QAM dapat dibentuk dengan menjumlahkan sebuah sinyal kosinus dengan amplitudo I(t) dan sebuah sinyal sinus dengan amplitudo Q(t). Ini sama dengan menjumlahkan sebuah sinyal AM (*amplitude modulation*) yang menggunakan *carrier* kosinus dengan sebuah sinyal AM lain yang menggunakan *carrier* sinus. Kata *quadrature* pada QAM berasal dari kedua carrier yang berbeda fase 90^0.

Amplitude dan *fase* untuk masing-masing simbol pada QAM dapat digambarkan dalam sebuah diagram dua dimensi yang disebut sebagai diagram konstelasi, seperti misalnya diagram konstelasi untuk 16-QAM yang dapat dilihat pada Gambar 2.7.



Gambar 2.7 Diagram Konstelasi 16 QAM

Sumbu x merupakan sumbu yang mewakili $\cos\omega_c$ dari persamaan dan disebut I (*imphase*), sedangkan sumbu y adalah sumbu yang mewakili $-\sin\omega_c$ dari persamaan dan disebut sumbu Q (*quadrature*). Data yang akan dikirim dibagi menurut jumlah bit untuk satu simbol. Setelah itu, data yang telah dibagi dipetakan menurut diagram konstelasi dengan menggunakan *mapper*. Keluaran *mapper* adalah komponen *imphase* dan *quadrature* untuk simbol yang ditentukan oleh data tadi. Kedua komponen ini yang akan diterukan pada proses selanjutnya.

Untuk M-ary *square* atau *rectanguler* signal dapat dituliskan persamaannya

$$s_i(t) = I_i\sqrt{\frac{Eo}{Ep}}p(t)\cos 2\pi fct - Q_i\sqrt{\frac{Eo}{Ep}}p(t)\sin 2\pi fct$$

$$= I_i\sqrt{\frac{Eo}{2}}\phi_1(t) + Q_i\sqrt{\frac{Eo}{2}}\phi_2(t) \qquad (2.5)$$

Dengan Eo adalah energi dari signal dengan amplitudo paling rendah, dan (Ii,Qi) adalah sepasang dari integer yang berdiri sendiri dimana penentuan lokasi dari signal point dalam konselasi. Maksimum nilai dari (Ii,Qi) adalah $(\pm 1,\pm 1)$. Sepasang (Ii,Qi) merupakan elemen dari matrik L x L seperti di bawah ini:

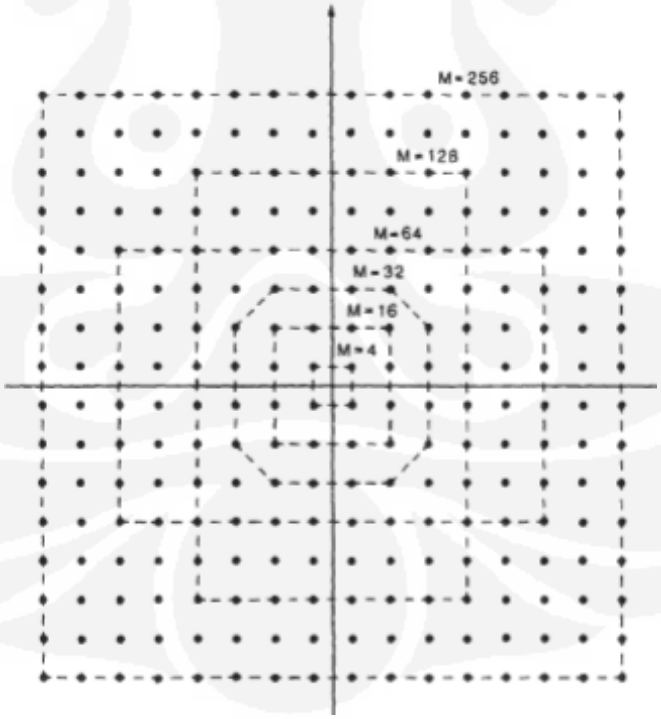
$$[I_i, Q_i] = \begin{bmatrix} (-L+1, L-1) & (-L+3, L-1) & \cdots & (L-1, L-1) \\ (-L+1, L-3) & (-L+3, L-3) & \cdots & (L-1, L-3) \\ \vdots & \vdots & & \vdots \\ (-L+1, -L+1) & (-L+3, -L+1) & \cdots & (L-1, -L+1) \end{bmatrix} \quad (2.6)$$

Dengan $L = \sqrt{M}$ $M = 2^n$ n= 1,2,3,...

Sebagai contoh untuk 16 QAM pada Gambar 2.7, dengan L=4 maka:

$$[I_i, Q_i] = \begin{bmatrix} (-3,3) & (-1,3) & (1,3) & (3,3) \\ (-3,1) & (-1,1) & (1,1) & (3,1) \\ (-3,-1) & (-1,-1) & (1,-1) & (3,-1) \\ (-3,-3) & (-1,-3) & (1,-3) & (3,-3) \end{bmatrix} \quad (2.7)$$

Gambar 2.8 menunjukan diagram konstelasi dari beberapa M-QAM.



Gambar 2.8 Diagram Konstelasi dengan beberapa M-QAM

2.9 *Transformasi Fourier*

Transformasi *Fourier* merupakan salah satu jenis transformasi yang digunakan untuk merubah ranah (*domain*) dari suatu sinyal, baik dari *domain* waktu ke *domain* frekuensi ataupun sebaliknya.

Hubungan antara *subcarrier-subcarrier orthogonal* yang digunakan dalam OFDM dapat diimplementasikan menggunakan transformasi *Fourier*, di mana pada sisi pemancar OFDM (*modulator*) menggunakan *Inverse Fast Fourier Transform* (IFFT) sedangkan pada sisi penerima OFDM (*demodulator*) menggunakan *Fast Fourier Transform* (FFT).

Transformasi *Fourier* diperlukan untuk menjaga *orthogonalitas subcarrier*, pada sisi pemancar dan penerima. Formula yang digunakan untuk FFT dan IFFT adalah.

IFFT $$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(n) e^{j2\pi kn/N} \qquad k = 0,1,2,..,N-1 \tag{2.8}$$

FFT $$x(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \qquad n = 0,1,2,..,N-1 \tag{2.9}$$

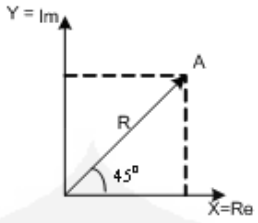
Kita perhatikan bahwa komputasi masing-masing titik pada IFFT dapat diselesaikan dengan perkalian kompleks N dan penambahan kompleks (N-1). Karena itu harga-harga IFFT N-titik dapat dihitung dalam total perkalian kompleks N^2 dan penambahan kompleks N(N-1).

Hal tersebut instruktif untuk memandang IFFT dan FFT sebagai transformasi linier pada berturut-turut barisan $\{x(n)\}$ dan barisan $\{X(k)\}$.

Berdasarkan teorema *Euler*, dapat dinyatakan bahwa :

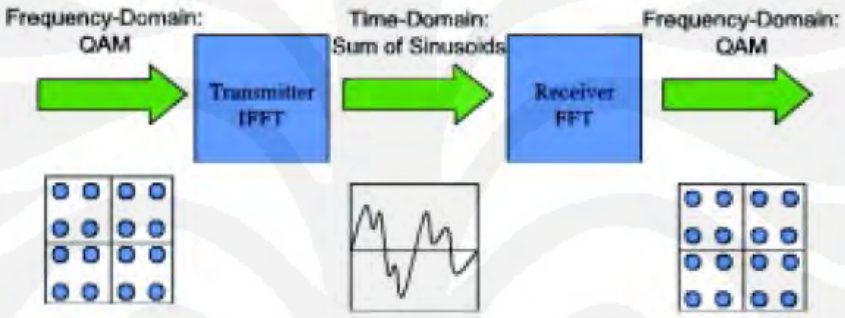
$$e^{j2\pi f_n t} = \cos(2\pi f_n t) + j\sin(2\pi f_n t) \tag{2.10}$$

Secara analitis, harga dari proses transformasi fourier dapat dilihat sebagai suatu bentuk kompleks yang mempunyai nilai *real (x)* dan *imajiner (y)*. Telah diketahui bahwa bentuk kompleks dapat juga dinyatakan dalam bentuk *polar*, yang terdiri dari *magnitude (R)* dan *phase (θ)*. Hubungan yang menggambarkan antara bentuk *polar* dan *rectangular* dapat dilihat pada Gambar 2.9.



Gambar 2.9 Hubungan Bentuk *Polar* dan *Rectangular*

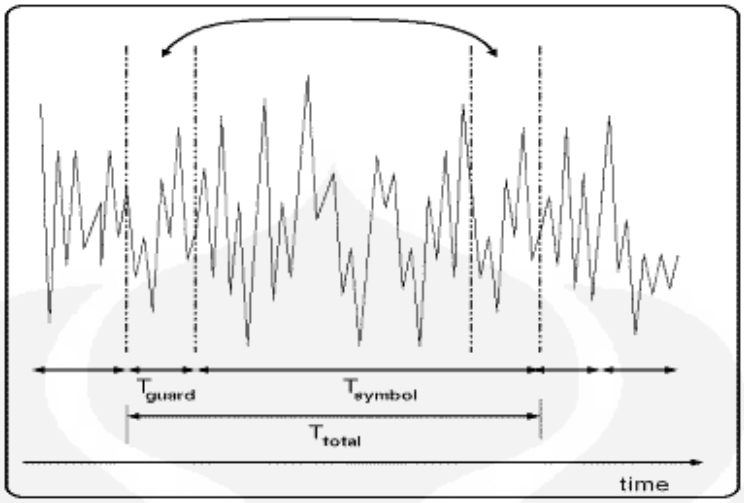
Berikut ini adalah diagram blok IFFT dan FFT pada sistem OFDM yang ditunjukan pada Gambar 2.10.



Gambar 2.10 Diagram Blok IFFT dan FFT pada Sistem OFDM

2.10 *Guard interval (Cycle prefix)*

Salah satu keunggulan sistem OFDM adalah ketahanan terhadap *multipath delay spread*, hal ini dapat dicapai dengan memiliki durasi simbol yang panjang yang akan meminimalisasi efek *delay spread*. Untuk memudahkan proses demodulasi pada bagian FFT di *receiver*, tiap-tiap subkanal OFDM haruslah terjaga orthogonalitasnya. Tetapi akibat respon kanal, akan terjadi *distorsi linear* yang menyebabkan energi pada tiap-tiap sub kanal menyebar ke sub kanal di sekitarnya. *Delay spread* menyebabkan waktu kedatangan sinyal bervariasi. Hal-hal inilah yang menyebabkan terjadinya *inter symbol interference* (ISI). Pendekatan yang digunakan untuk mengatasi masalah tersebut adalah penambahan *guard interval* atau biasa disebut dengan *cycle prefix*. Gambar dibawah ini memvisualisasikan penambahan *guard interval*, yaitu bagian akhir dari satu simbol OFDM di-*copy* dan diletakkan di awal simbol seperti ditunjukan pada Gambar 2.11.



Gambar 2.11 Penambahan *Guard Interval* atau *Cycle Prefix*

Dengan adanya *guard interval* maka durasi simbol OFDM terkirim akan lebih panjang, total durasi simbol OFDM terkirim (OFDM *block length*) menjadi:

$$T\ total = T\ guard + T\ symbol \qquad (2.11)$$

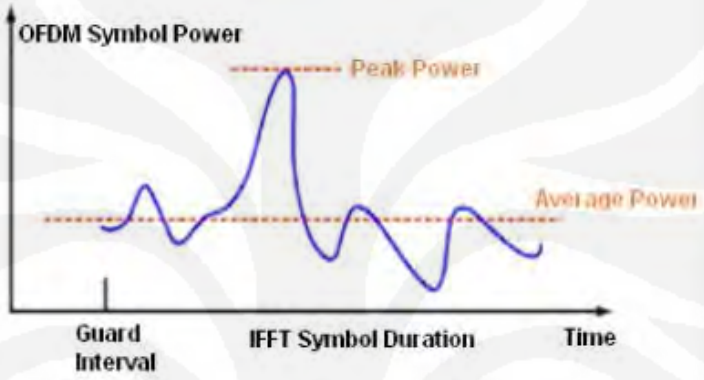
Walaupun terdapat alokasi *guard interval*, simbol OFDM terkirim masih akan tetap terkena interferensi antar simbol. Tetapi bagian simbol OFDM yang terinterferensi tersebut adalah *guard interval* dari simbol OFDM itu sendiri. Bagian symbol OFDM yang mengandung sinyal informasi tidak terkena interferensi. Pada *receiver*, *guard interval* akan dihilangkan sehingga rekonstruksi sinyal OFDM akan meminimalisasi terjadinya *error*.

2.11 PAPR (*Peak to Average Power Ratio*)

Salah satu kendala dalam sistem OFDM adalah nilai PAPRnya yang tinggi. PAPR adalah perbandingan antara daya puncak sinyal dengan daya rata-ratanya. Perbandingan nilai PAPR dengan kualitas sinyal berbanding terbalik, yaitu semakin besar harga PAPR, kualitas sinyal semakin menurun. Begitu pula sebaliknya makin kecil PAPR, kualitas sinyal semakin baik. Idealnya nilai puncak daya suatu sinyal sama dengan nilai daya rata ratanya.

PAPR merupakan ukuran dari fluktuasi tepat sebelum *amplifier*. Misal PAPR sinyal hasil dari *mapping* PSK *base band* sebesar 0 dB karena semua

symbol mempunyai daya yang sama. Tetapi setelah dilakukan proses IDFT/IFFT, hasil superposisi dari dua atau lebih *subcarrier* dapat menghasilkan variasi daya dengan nilai *peak* yang besar. Hal ini disebabkan oleh modulasi masing-masing *subcarrier* dengan frekuensi yang berbeda sehingga apabila beberapa *subcarrier* mempunyai fasa yang koheren, akan muncul amplitudo dengan level yang jauh lebih besar dari daya sinyalnya. Gambar 2.12 menunjukan *Peak power* sinyal dengan rata-rata *power* dari sistem OFDM.



Gambar 2.12 Peak Power dan Average Power pada Sistem OFDM

Peak to average power ratio dapat dinyatakan dalam persamaan matematis sebagai berikut :

$$PAPR = \frac{\max\left|s(n)^2\right|}{E\left[\left|s(n)^2\right|\right]}, 1 \le n \le N \tag{2.12}$$

Dengan:

$\max|s(n)^2|$ adalah nilai maksimum daya

$E[|s(n)^2|]$ adalah nilai rata-rata daya

Nilai PAPR yang tinggi memiliki beberapa efek negatif yang tidak dapat diabaikan sehingga diperlukan suatu teknik untuk mereduksinya sehingga dapat mengurangi degradasi performansi OFDM dan mengurangi besarnya nilai IBO sehingga performansi OFDM dan efisiensi penggunaan PA meningkat. Berbagai teknik reduksi PAPR telah dikembangkan dan dapat dikelompokkan dalam dua kategori, yaitu :

1. *Signal distortion,* seperti *Clipping,* dan
2. *Symbol scrambling,* seperti *Selective Mapping* (SLM), *Partial Transmit Sequence* (PTS), *Coding* (Golay Complementary codes) dan *Dummy Sequence Insertion* (DSI).

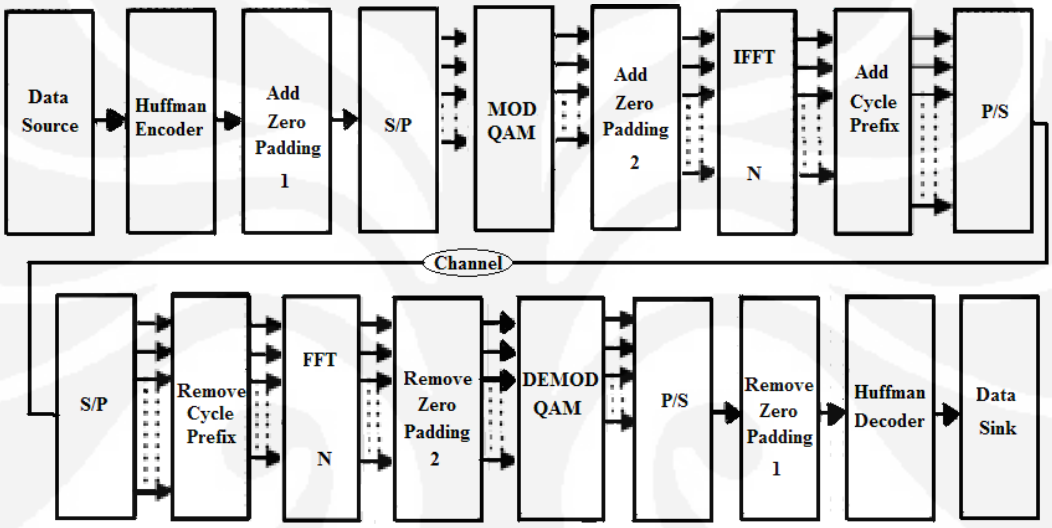
Kerugian yang ditimbulkan oleh tingginya *PAPR* antara lain:

1. *Intermodulation* diantara *subcarrier.*
2. Besarnya penggunaan *bandwidth.*
3. Pemotongan puncak sinyal atau *Clipping.*

BAB 3
PERANCANGAN MODEL DAN SIMULASI SISTEM

3.1 Pemodelan Sistem

Simulasi pada skripsi ini dibuat untuk menganalisa kinerja *Huffman Coding* pada sistem OFDM. Berikut ini adalah diagram blok sistem utama ditunjukan pada Gambar 3.1.



Gambar 3.1 Diagram Blok Sistem OFDM dengan *Huffman Coding*

Gambar 3.1. menunjukkan diagram blok simulasi sistem OFDM terhadap pengaruh *Huffman Coding*. Proses simulasi diawali dengan proses *Huffman Encoder* untuk mengkompres data *source* (*random source*), dan dilanjutkan dengan proses *add zero padding* 1 dan proses *serial to paralel,* kemudian data dimodulasi dengan menggunakan QAM *modulation,* setelah itu proses *add zero padding* 2, dan dilanjutkan dengan proses IFFT, proses *add cycle prefix,* proses *parallel to serial,* dan kanal. Kanal dalam simulasi tidak dipengaruhi oleh AWGN. Setelah melalui kanal proses selanjutnya dengan proses *serial to parallel,* proses *remove cycle prefix,* proses FFT, proses *remove zero padding* 2, proses QAM *demodulation, ,* proses *parallel to serial,* proses *remove zero padding* 1, dan proses *Huffman Decoder*. Kemudian data s*ink* akan menghasilkan data yang sudah didekompresi oleh *Huffman decoder*.

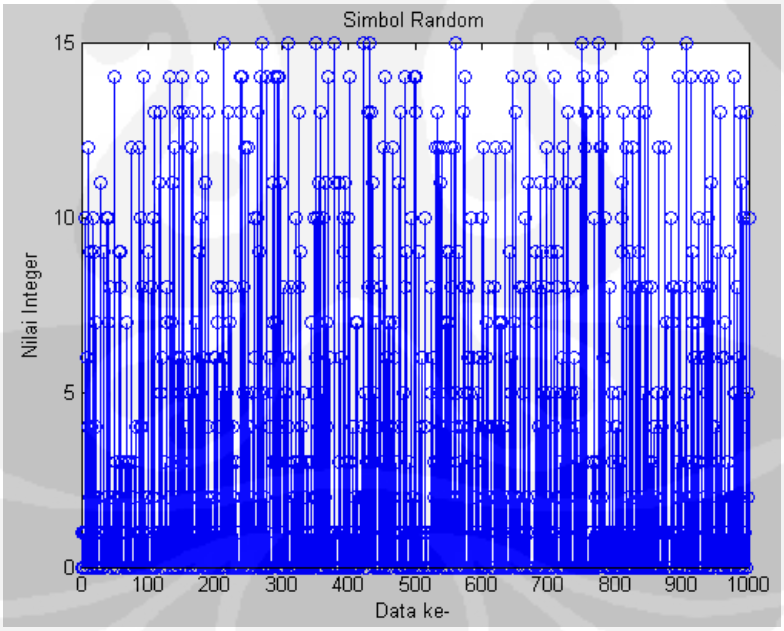
3.2 Parameter Simulasi

Dalam skripsi ini parameter-parameter yang digunakan adalah sebagai berikut:

1. Data *source* yang dibangkitkan sebesar 6720.
2. Modulasi yang digunakan adalah 16 QAM
3. Jumlah *subcarrier* yang digunakan adalah 192.
4. Panjang IFFT adalah 256.
5. Penambahan *cycle prefix* sebesar ¼ dari panjang IFFT
6. Pengkombinasian sistem dengan *Huffman Coding.*

3.3 Data *source*

Dalam simulasi ini, informasi dibangkitkan secara acak atau *random* dengan jumlah data yang dibangkitkan adalah sebanyak 6720 dalam bentuk integer (0-15), Berikut ini adalah Gambar 3.2 sebagai contoh jumlah informasi dari random integer dengan bilangan 0-15 yang dibangkitkan sebanyak 1000 data.



Gambar 3.2 Jumlah Informasi Random Integer

Untuk mengetahui nilai data yang dibangkitkan sebanyak 1000 data pada Gambar 3.2 dapat dilihat pada lampiran data random yang dibangkitkan sebanyak 1000 data.

Data *source* ini dibuat dengan perbandingan antara nilai integer antara satu dengan yang lain memiliki probabilitas yang *significant.* Hal ini bertujuan agar input yang akan masuk ke blok *Huffman Encoder* harus memiliki nilai probabilitas yang berbeda jauh antara satu dengan yang lainnya, karena pada *Huffman Encoder* akan membentuk bit-bit pengganti dari data originalnya.

Pada simulasi ini, dapat memanfaatkan fungsi *rand* pada matlab dan pengkombinasian matematik untuk mendapatkan data random yang memiliki probabilitas yang *significant* antara satu dengan yang lainnya. Nilai prosentase dapat dihitung menggunakan persamaan 3.1.

$$P = \frac{N}{T} \tag{3.1}$$

Dengan:

P adalah prosentasi (%)

N adalah banyak data yang keluar

T adalah total data yang dibangkitkan

Prosentase jumlah output 0-15 yang dibangkitkan sebanyak 1000 data berdasarkan Gambar 3.2 ditunjukkan pada Tabel 3.1.

Tabel 3.1 Prosentase jumlah informasi

Nilai Integer	Prosentasi
0	0,320
1	0,153
2	0,089
3	0,053
4	0,052
5	0,046
6	0,037
7	0,033
8	0,043
9	0,027
10	0,020
11	0,022
12	0,026
13	0,029
14	0,030
15	0,012

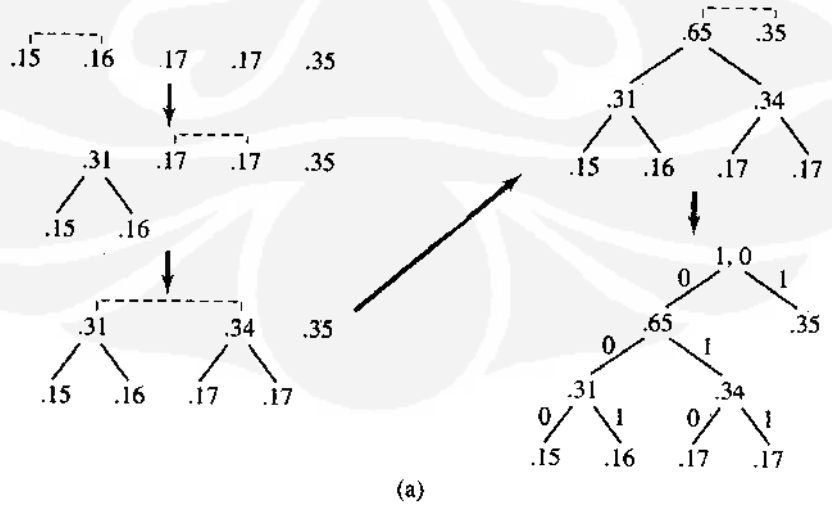
3.4 Proses *Huffman Coding*

Huffman coding menggunakan metode spesifik untuk merepresentasikan setiap simbol yang menghasilkan suatu kode prefix. Kode prefix ini merupakan sekumpulan kode biner yang pada kode ini tidak mungkin terdapat kode prefix yang menjadi awalan bagi kode biner yang merepresentasikan simbol lain, sehingga dalam pembentukan kode huffman harus unik. Hal ini akan mencegah timbulnya keraguan dalam proses *decoding.* Dalam kode Huffman, kode biner untuk simbol dengan kekerapan lebih besar akan memiliki kode yang lebih pendek daripada untuk simbol dengan kekerapan lebih kecil.

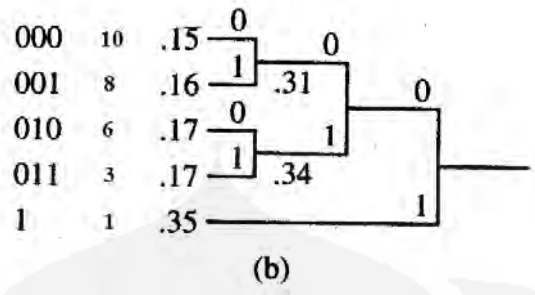
3.4.1 Proses *Huffman Encoder*

Aliran data input yang berasal dari data *source* akan masuk ke *Huffman Encoder* Pada proses *Huffman Encoder* dapat dibentuk dengan membuat pohon *Huffman* dengan cara mengurutkan simbol-simbol dari probabilitas yang terbesar sampai probabilitas yang terkecil untuk mendapatkan probabilitas yang baru dan urutkan kembali dari yang terbesar sampai yang terkecil. Begitu seterusnya sampai mendapatkan jumlah probabilitas sama dengan satu. Selanjutnya berikan kode bit 1 untuk sebelah kanan dan kode bit 0 untuk sebelah kiri.

Sebagai contoh sederhana pembentukan pohon *Huffman* dalam skripsi ini adalah ketika simbol data yang akan dikirim dari data source yaitu 1, 3, 6, 8, 10 dengan probabiltas masing-masing nilainya adalah 0,35; 0,17; 0,17; 0,16; 0,15. Proses pembentukan *Huffman Coding* dapat dilhat pada Gambar 3.3.



(a)



Gambar 3.3 Ilustrasi Pembentukan *Huffman Coding*

Gambar 3.3 memperlihatkan bahwa simbol 1 memiliki probabilitas yang paling besar sehingga memiliki kode *Huffman* yang paling sedikit, sedangkan untuk simbol 2, 6, 8, 10 memiliki probabilitas yang lebih kecil daripada sibol 1 sehingga memiliki kode *Huffman* yang lebih besar dari probabilitas yang paling kecil. Gambar 3.3 dapat dibuat tabel kode *Huffman*-nya seperti ditunjukan pada Tabel 3.2.

Tabel 3.2 Tabel Perbandingan Kode Original dengan Kode *Huffman*

Nilai Integer	Bit Original	Bit Huffman
1	0001	1
3	0011	011
6	0110	010
8	1000	001
10	1010	000

Pada Tabel 3.2 dapat dilihat terjadi pengompresan data dari 4 bit menjadi 3 bit dan dari 4 bit menjadi 1 bit, sehingga ketika data tersebut akan dikirim maka terjadi penghematan bit. Jika data tersebut tidak dikompres menggunakan *Huffman Coding* maka data yang akan dikirim sebesar

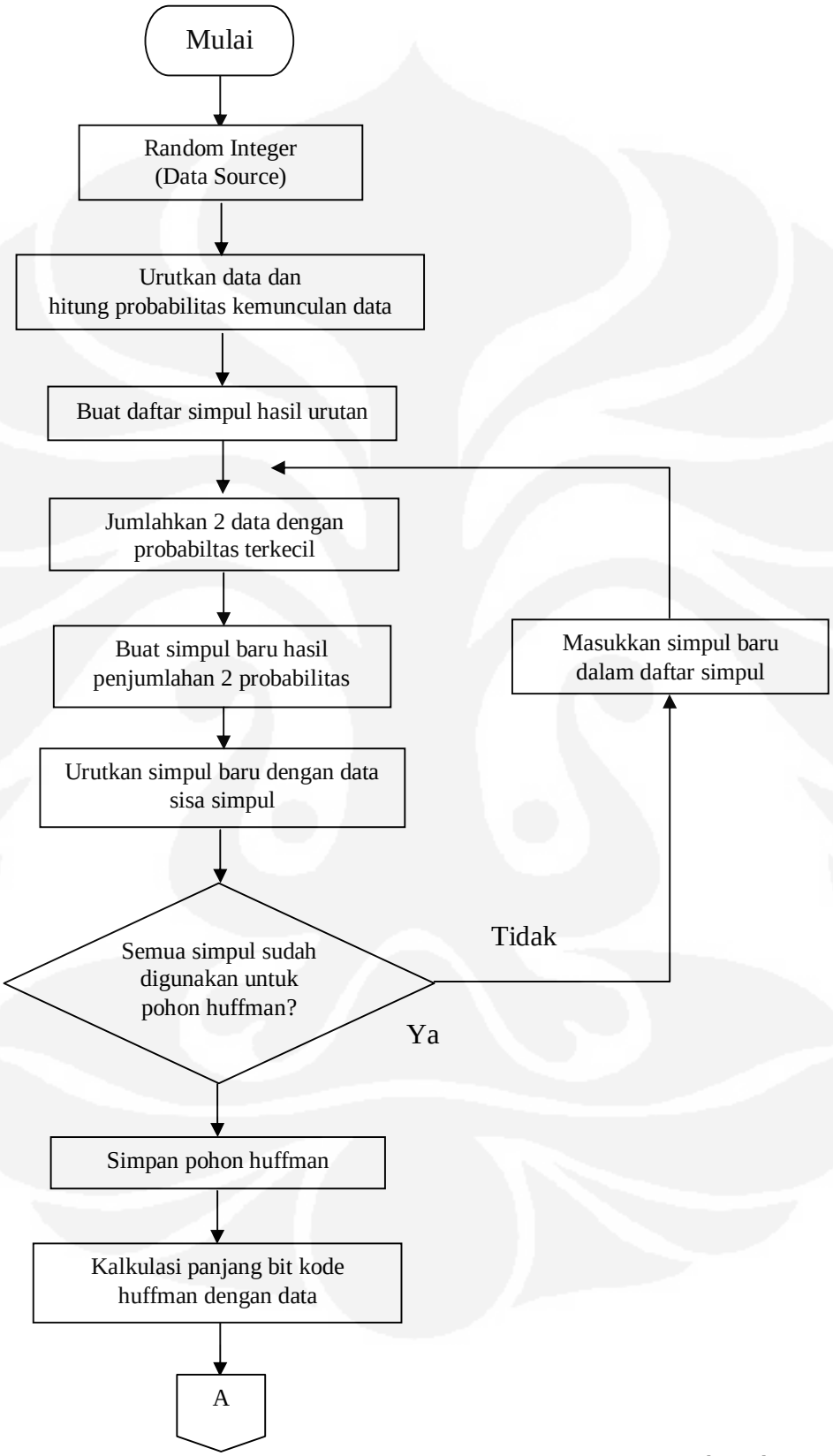
Data normal = 5 x 4 = 20 bit,

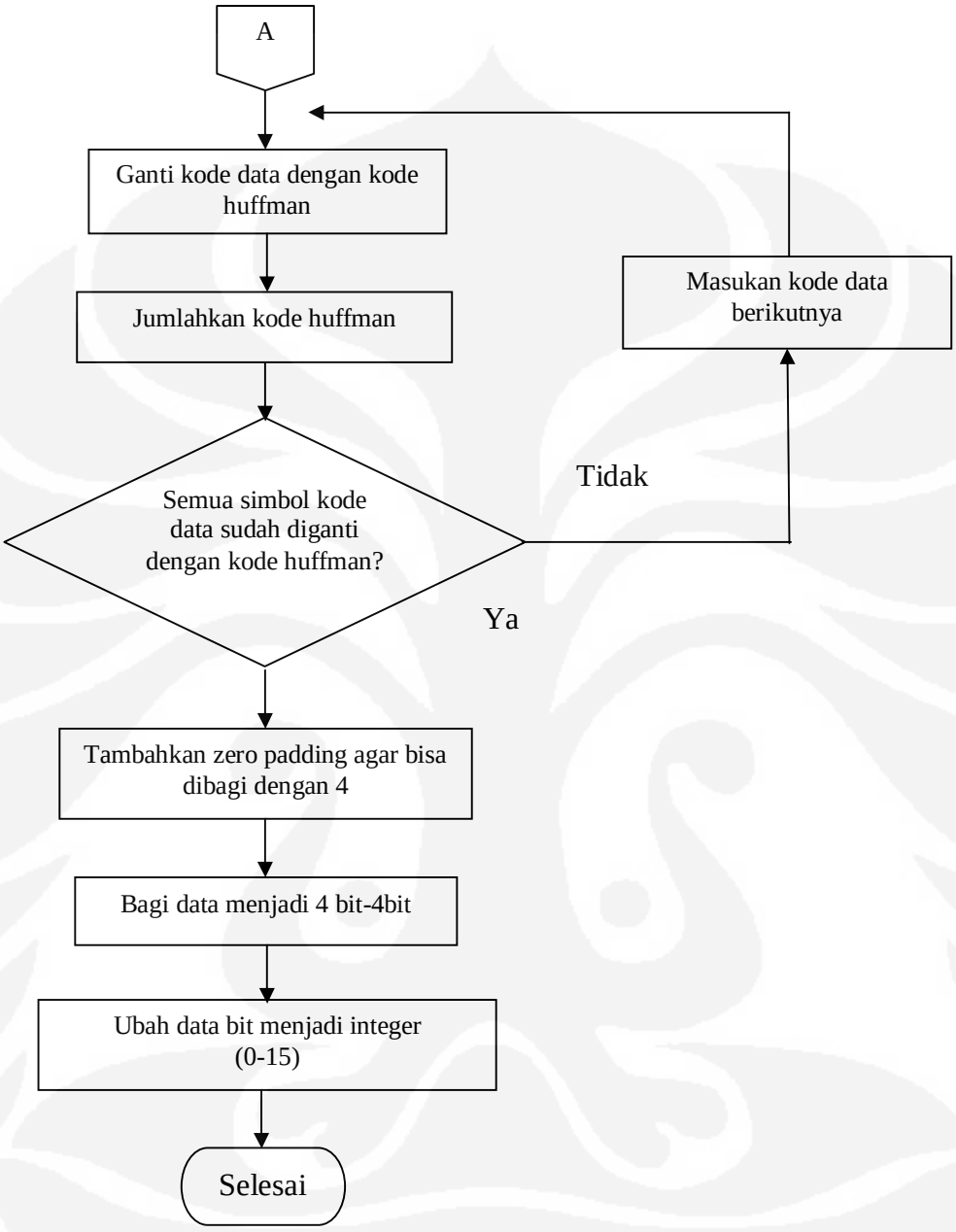
sedangkan jika data tersebut akan dikirim dengan menggunakan *Huffman Coding*, maka data yang dikirim menjadi

Data *Huffman* = (3x4) + (1x1) = 12 +1 = 13 bit.

Terjadi penghematan bit sebesar 20 -13 = 7 bit.

Berikut ini adalah *flowchart* algoritma *Huffman Encoder* yang ditunjukan pada Gambar 3.4.



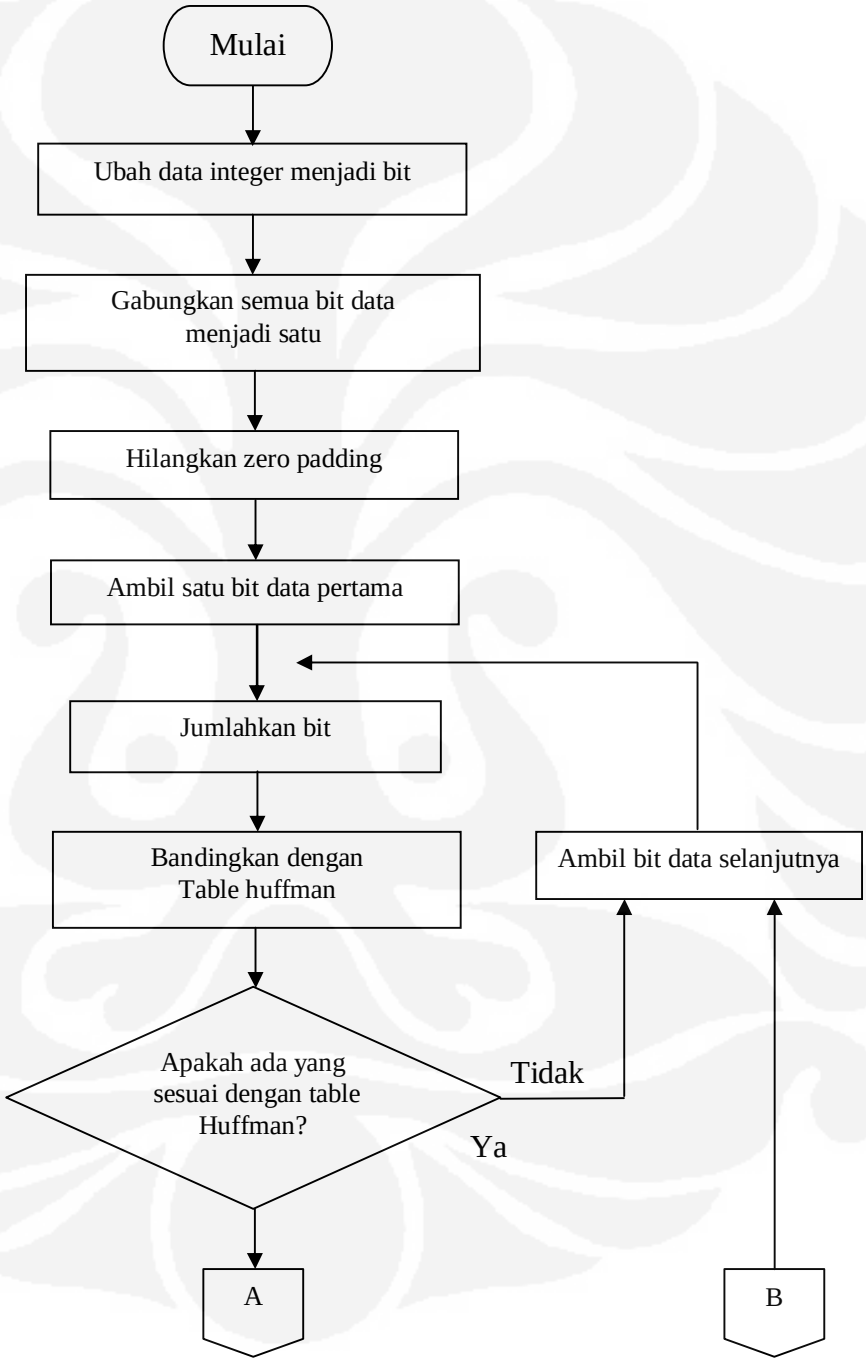


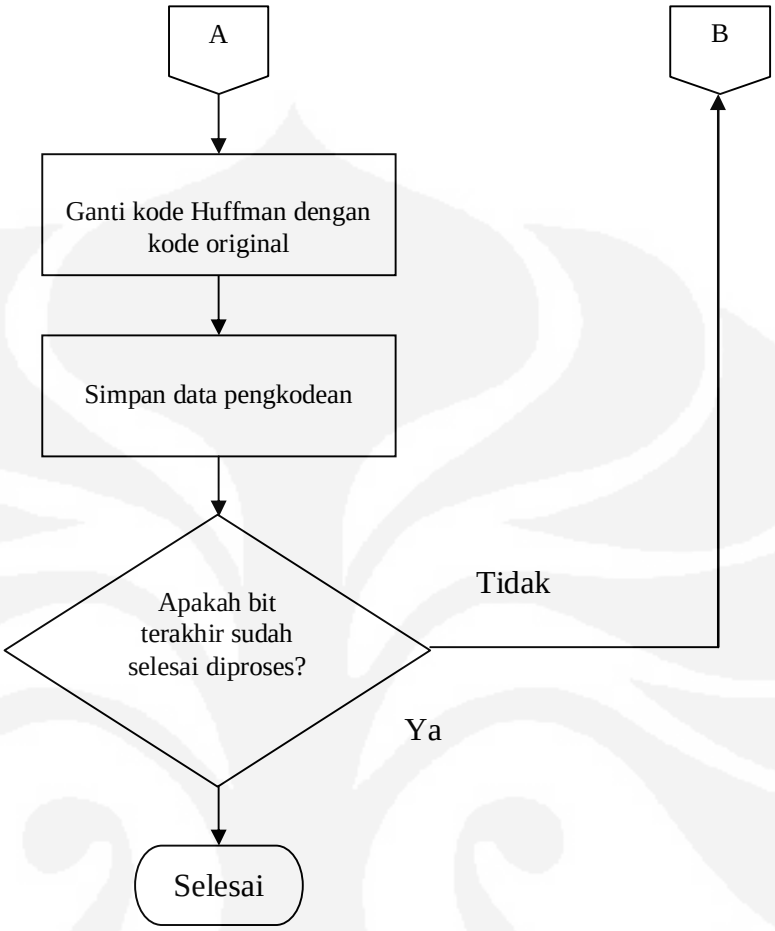
Gambar 3.4 *Flowchart* Algoritma *Huffman Encoder*

3.4.2 Proses *Huffman Decoder*

Proses *Huffman Decoding* adalah sebuah proses untuk menyusun kembali data yang telah dikodekan sebelumnya sehingga informasi yang diterima dapat dibaca dan diolah. Penguraian kode (*Decoding*) ini adalah lawan dari pengkodean

(*Encoding*). Ada 2 cara penguraian kode *Huffman.* Cara yang pertama adalah menggunakan pohon *Huffman* sedangkan cara yang kedua adalah menggunakan tabel *Huffman Coding*. Dalam skripsi ini menggunakan cara tabel *Huffman Coding*. Berikut ini adalah *flowchart* algoritma dari *Huffman Decoder* yang ditunjukan pada Gambar 3.5.





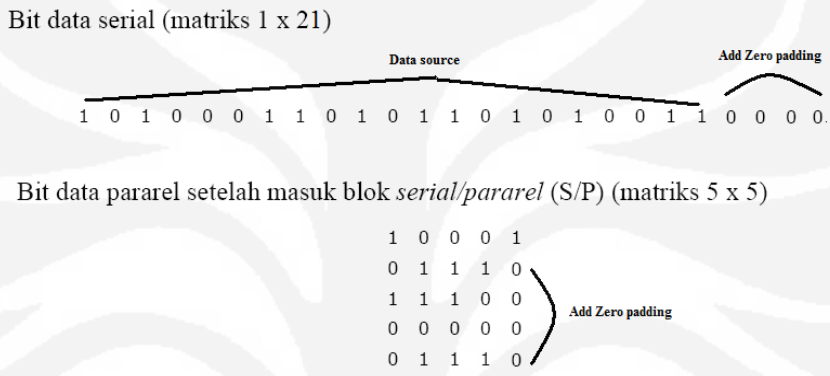
Gambar 3.5 *Flowchart* Algoritma *Huffman Decoder*

3.5 Zero padding

Zero padding adalah penambahan data berupa nilai nol, tetapi tidak menghasilkan informasi data baru atau *zero padding* disebut juga data *dummy*. Berdasarkan Gambar 3.1 bahwa *zero padding* diberikan sebelum proses *serial to parallel* dan sebelum proses IFFT.

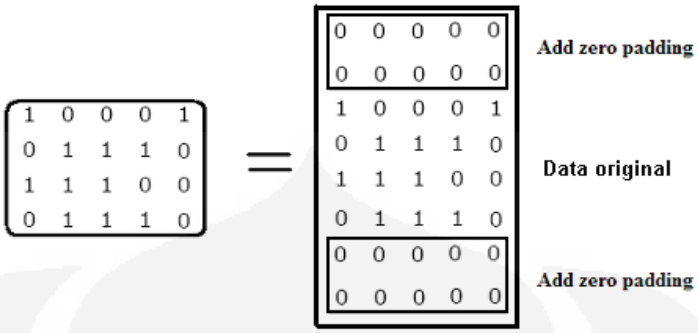
Zero padding yang diletakkan sebelum proses *serial to parallel* digunakan sebagai pelengkap pembagian berdasarkan nilai *subcarrier* atau data yang diparalelkan karena proses kompresi data yang menggunakan *Huffman Coding* menyebabkan data yang akan ditransmisikan mengalami penurunan dalam jumlah bitnya dan tidak selalu bisa diparalelkan sesuai dengan jumlah *subcarrier* yang diinginkan. Untuk itu dibutuhkan sejumlah data nol agar data hasil kompresi

dengan menggunakan *Huffman Coding* dapat diparalelkan sesuai dengan jumlah *subcarrier*. Berikut ini adalah Gambar 3.6 yang menunjukan penambahan *zero padding* sebelum proses *serial to parallel*, dan terjadi saat data masih dalam bentuk serial. Pada Gambar 3.6 menunjukan bahwa data *source* yang dibangkitkan sebesar 21 data, dan jika dalam matrik sebesar 1 x 21. Tetapi proses *serial to parallel* membutuhkan matrik sebesar 5 x 5 sehingga dibutuhkan 4 data lagi agar data tersebut dapat masuk ke proses *serial to parallel*.



Gambar 3.6 Penambahan *Zero Padding* sebelum Proses *Serial to Parallel*

Sedangkan *zero padding* yang diletakkan sebelum proses IFFT, bertujuan untuk memperbaiki sinyal PAPR sehingga menghasilkan *performance* yang lebih baik. Berikut ini adalah Gambar 3.7 yang menunjukan penambahan *zero padding* sebelum proses IFFT. Pada Gambar 3.7 menunjukan bahwa penambahan data terjadi pada saat data dalam keadaan paralel dan kemudian disisipkan nilai nol di awal dan diakhir sampai dengan penjumlahan data *subcarrier* dan data nol sama dengan jumlah IFFT. Misalkan jumlah data *source* dalam matrik adalah 4 X 5, sedangkan jumlah IFFTnya atau matrik saat proses IFFT sebesar 8 X 5, sehingga diperlukan penambahan nilai nol dengan matrik sebesar 2 X 5 di awal dan di akhir.

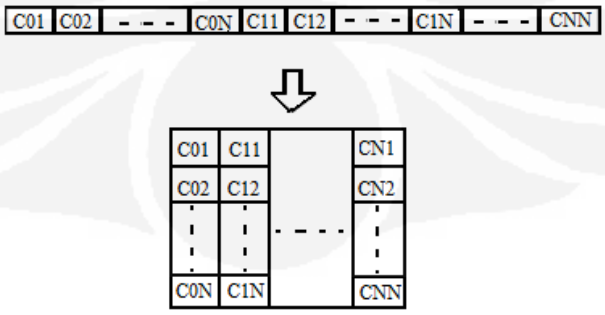


Gambar 3.7 Penambahan *Zero Padding* sebelum Proses IFFT

Pada proses *remove zero padding* merupakan proses penghilangan bit-bit bernilai nol yang ditambahkan sebelum proses *serial to parallel* dan sebelum proses IFFT pada *tranmitter*. Hal ini bertujuan agar data yang akan masuk ke *Huffman Decoder* sama dengan data output *Huffman Encoder* sehingga data tersebut dapat diterjemahan sesuai data pengirim.

3.6 Serial to Parallel Converter

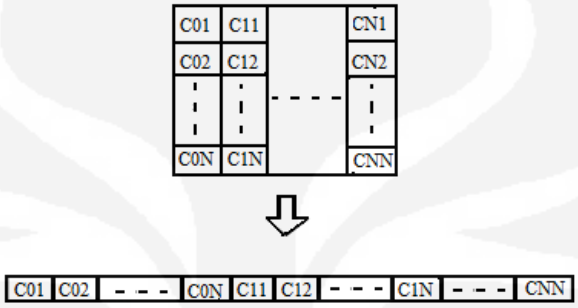
Input dari *Serial to Parallel Converter* adalah proses mengubah data dari serial menjadi paralel. Pengiriman data dilakukan setiap sc simbol, di mana sc merupakan jumlah subkanal = jumlah *subcarrier*. Pada pemodelan ini digunakan 192 subkanal (sc = 192) sehingga jika dimisalkan sc simbol pertama adalah x[1],x[2],…,x[N], maka pada proses S/P *converter* ini simbol x[1] dikirimkan melalui subkanal pertama, x[2] dikirimkan melalui subkanal ke-2 dan seterusnya hingga x[N] dikirimkan melalui subkanal ke-sc. Proses *serial to parallel* ditunjukkan seperti Gambar 3.8. Pada simulasi, pemodelan konsep ini cukup diwakili dengan pemakaian fungsi *reshape* dari Matlab.



Gambar 3.8 *Serial to Pararrel Converter*

3.7 Parallel to Serial Converter

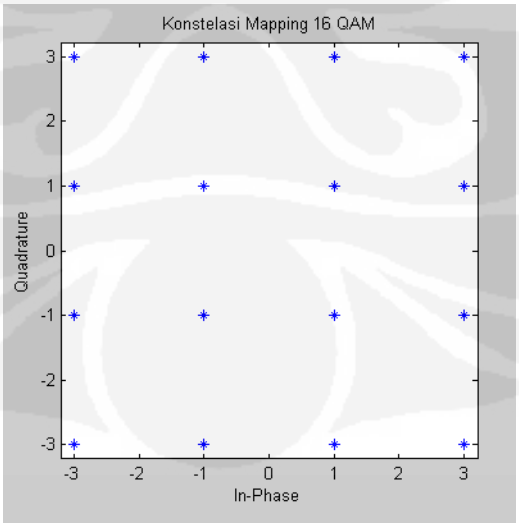
Hasil proses IFFT diserialkan terlebih dahulu sebelum masuk ke kanal, dengan menggunakan *Parallel to Serial Converter*, sedangkan pada Matlab cukup menggunakan fungsi *reshape*. Hasil dari *reshape* selanjutnya akan ditransmisikan melalui kanal. Pada Gambar 3.9, nampak bahwa data yang sebelumnya terbagi dalam N buah subkanal, akan diatur kembali menjadi deretan data serial.



Gambar 3.9 *Pararrel to Serial Converter*

3.8 Proses modulasi dan demodulasi QAM

Pada modulasi QAM, setiap bit data yang masuk akan didistribusikan pada kanal *Inphase* (I) dan *Quadrature* (Q). Setelah *bit-bit* tersebut dipetakan ke kanal I dan Q, kemudian diberikan level simbol. Dengan mengasumsikan kanal *inphase* untuk simbol real dan kanal *quadrature* untuk imajiner maka akan terdapat 16 titik konstelasi dari hasil modulasi 16 QAM. Gambar 3.10 menunjukkan konstelasi modulasi 16 QAM menggunakan pemrograman Matlab.



Gambar 3.10 Konstelasi Sinyal 16 QAM

Untuk menggambarkan konstelasi dari hasil modulasi QAM yang sesuai dengan Gambar 3.10 di atas, maka dapat dibuat dengan menggunakan fungsi *scatterplot* dari Matlab, sedangkan untuk membentuk bilangan kompleks pada modulasi QAM dapat menggunakan fungsi *qammod* dari Matlab. Berikut ini adalah Tabel 3.3 yang menunjukkan pemetaan *bit input* ke dalam simbol 16 QAM.

Tabel 3.3 Pemetaan simbol 16 QAM

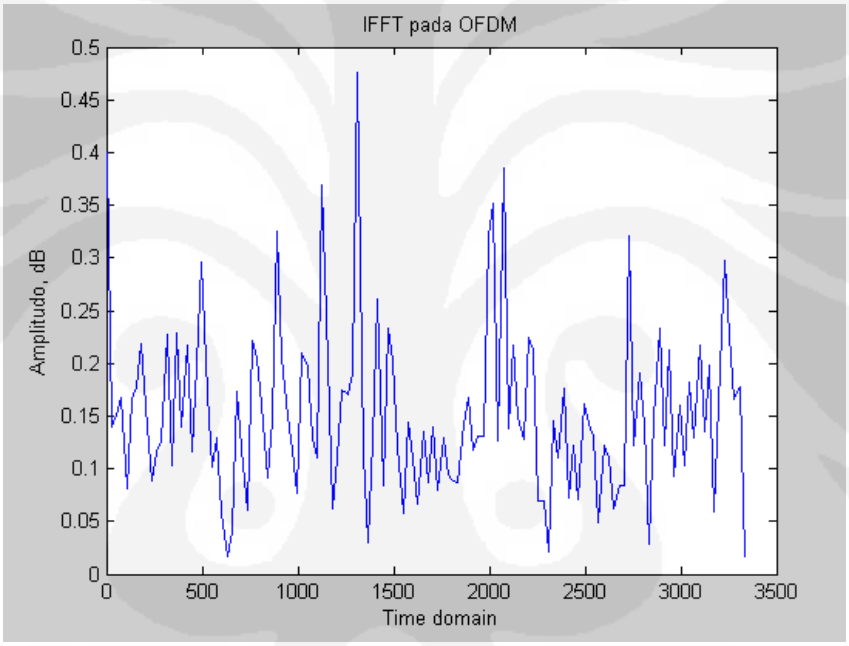
Integer	Bit	I	Q
0	0000	-3	3j
1	0001	-3	1j
2	0010	-3	-1j
3	0011	-3	-3j
4	0100	-1	3j
5	0101	-1	1j
6	0110	-1	-1j
7	0111	-1	-3j
8	1000	1	3j
9	1001	1	1j
10	1010	1	-1j
11	1011	1	-3j
12	1100	3	3j
13	1101	3	1j
14	1110	3	-1j
15	1111	3	-3j

Pada demodulasi QAM, dilakukan untuk memetakan kembali simbol ke dalam *bit - bit* informasi yang dimodulasi di pemancar. Simbol dipetakan kembali ke dalam bentuk *bit – bit* dimana dilakukan pendeteksian magnitudo dan fasa dari simbol tersebut.

Pada demodulasi QAM diperlukan proses untuk menggabungkan kembali sinyal dari kanal *Inphase* dan *Quadrature*, dengan cara memetakan simbol-simbol ke dalam bentuk *bit-bit* sesuai dengan diagram konstelasi QAM. Untuk mengembalikan proses modulasi atau dikenal dengan proses demodulasi dapat menggunakan fungsi *qamdemod* pada Matlab.

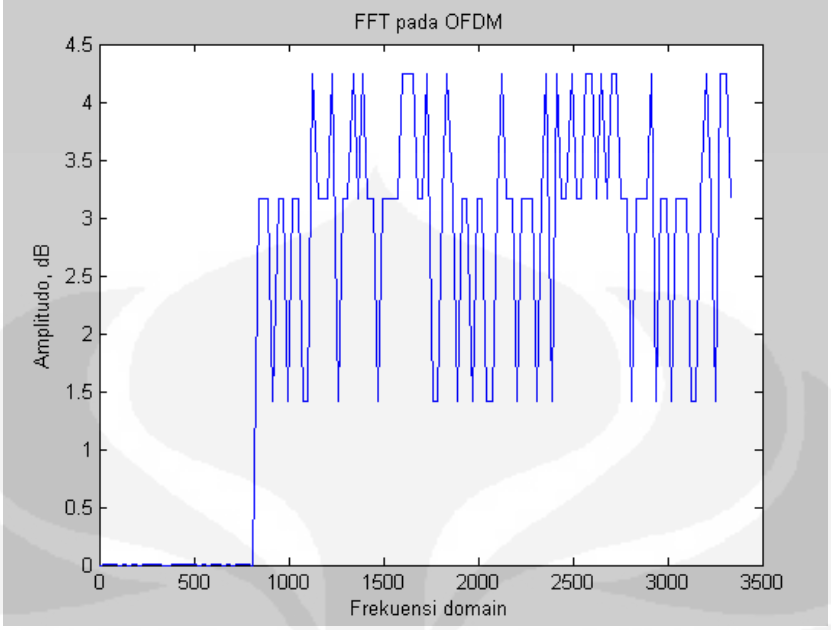
3.9 Proses IFFT dan FFT

Orthogonalitas *carrier* harus terjaga, sehingga pada *domain* frekuensi, masing-masing subkanal diijinkan untuk saling *overlapping*. Untuk menjamin orthogonalitas *carrier*, maka pada proses IFFT pada simulasi ini digunakan fungsi *ifft* dari Matlab, sehingga *output* dari IFFT merupakan bentuk sinyal OFDM dalam *domain* waktu. Dengan adanya proses IFFT, maka sinyal dapat saling *overlapping* tetapi tidak saling *interference*. Berikut ini adalah Gambar 3.11 menunjukan sinyal output IFFT pada sistem *transmitter* OFDM dengan menggunakan program Matlab.



Gambar 3.11 Sinyal output IFFT pada sistem *transmitter* OFDM

Proses FFT merupakan kebalikan dari proses IFFT, IFFT pada pemancar berfungsi menggabungkan sinyal dalam modulasi *multi-carrier*, sedangkan FFT berfungsi memisahkan kembali sinyal informasi dari sinyal *carrier* dalam rangkaian proses demodulasi pada teknik *multi-carrier*. Untuk implementasi dalam simulasi di Matlab cukup menggunakan fungsi *fft.* Berikut ini adalah Gambar 3.12 menunjukan sinyal output FFT pada sistem *receiver* OFDM dengan menggunakan program Matlab.



Gambar 3.12 Sinyal output FFT pada sistem *receiver* OFDM

3.10 Cycle prefix (Guard interval)

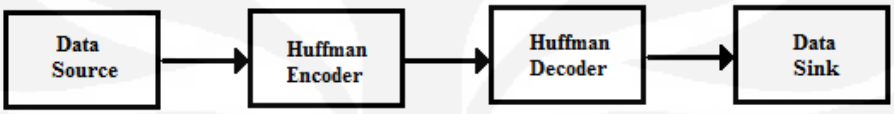
Pada bab 2 telah dijelaskan bahwa *cycle prefix* adalah penyisipan atau pengkopian sebagian data dengan tujuan untuk mengurangi terjadinya *inter symbol interference* (ISI). Dalam sistem ini dibuat penambahan data sebesar 25 % dari panjang simbol OFDM.

Pada proses *remove cycle prefix* merupakan pengembalian data setelah terjadi penyisipan atau pengkopian data yang dilakukan setelah proses IFFT yang terjadi di bagian *transmitter*. Pada proses ini jumlah data menjadi kembali sama dengan proses IFFT.

BAB 4
PENGUJIAN DAN ANALISA

4.1 Pengujian dan Analisa Program *Huffman Coding* tanpa OFDM

Dalam pengujian sistem *Huffman Coding,* akan dianalisa perbedaan jumlah bit original dengan bit *Huffman Coding* dari data yang akan dikompresi dan presentasi nilai kompresi yang dihasilkan oleh *Huffman Coding.* Sistem pengujian *Huffman Coding* tanpa OFDM ditunjukan pada Gambar 4.1.



Gambar 4.1 Pengujian *Huffman Coding* tanpa OFDM

Gambar 4.1 menunjukan bahwa sistem pengujian *Huffman Coding* tanpa OFDM terdiri dari data *source* sebagai pambangkit data *random* (data input), *Huffman Encoder* sebagai pengkompres data yang berasal dari data *source*, *Huffman Decoder* sebagai dekompres data dari hasil kompres oleh *Huffman Encoder*, dan data *sink* sebagai data output.

Untuk mengetahui nilai bit yang dihasilkan dari sistem *Huffman Coding*, maka dalam pengujian kali ini dilakukan dengan 2 tipe pembangkit probabilitasnya yaitu probabilitas dengan nilai perbedaan yang *significant* dan probabilitas dengan nilai perbedaan yang merata, dengan parameter pengujiannya adalah data yang dibangkitkan sebanyak 100 dan rentang nilai integernya adalah dari 0 sampai 15. Berikut ini adalah Tabel 4.1 menunjukan hasil pengujian kompresi *Huffman coding* dengan perbedaan nilai probabilitas yang *significant* dan Tabel 4.2 menunjukan hasil pengujian kompresi *Huffman Coding* dengan perbedaan nilai probabilitas yang merata.

Tabel 4.1 Hasil Pengujian Kompresi *Huffman Coding* dengan Nilai Probabilitas yang *Significant*

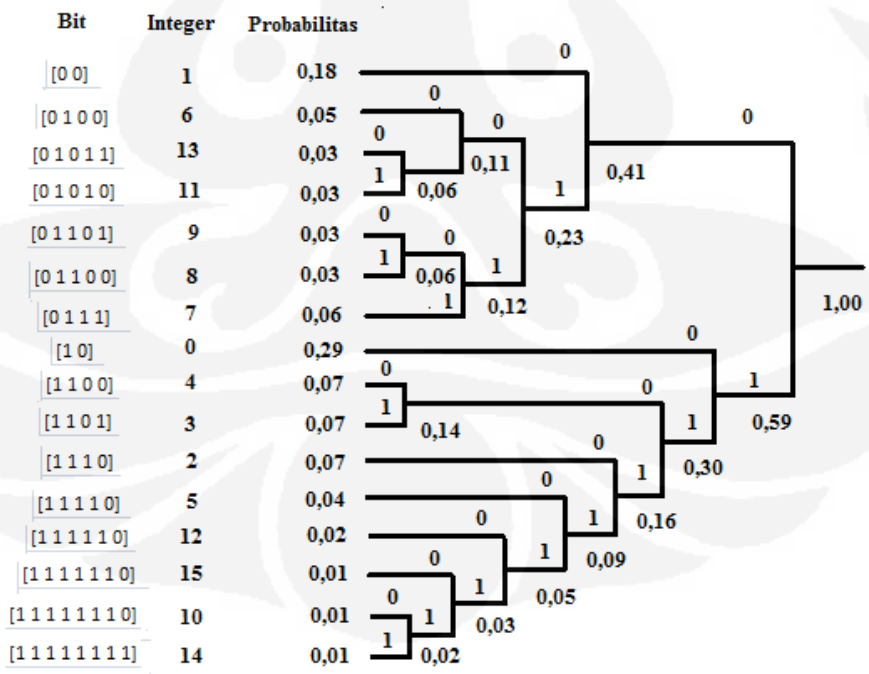
Nilai Integer	Probabilitas nilai	Bit original	Bit *Huffman coding*
0	0,2900	0000	1 0
1	0,1800	0001	0 0
2	0,0700	0010	1 1 0 0
3	0,0700	0011	1 1 0 1
4	0,0700	0100	1 1 1 0
5	0,0400	0101	1 1 1 1 0
6	0,0500	0110	0 1 0 0
7	0,0600	0111	0 1 1 1
8	0,0300	1000	0 1 1 0 0
9	0,0300	1001	0 1 1 0 1
10	0,0100	1010	1 1 1 1 1 1 1 0
11	0,0300	1011	0 1 0 1 0
12	0,0200	1100	1 1 1 1 1 0
13	0,0300	1101	0 1 0 1 1
14	0,0100	1110	1 1 1 1 1 1 1 1
15	0,0100	1111	1 1 1 1 1 1 0

Tabel 4.2 Hasil Pengujian Kompresi *Huffman Coding* dengan Nilai Probabilitas yang Merata

Nilai Integer	Probabilitas nilai	Bit original	Bit *Huffman coding*
0	0,050	0000	0 0 0 0
1	0,110	0001	0 1 1
2	0,050	0010	0 0 0 1
3	0,070	0011	1 0 1 0
4	0,050	0100	0 1 0 0
5	0,090	0101	1 1 1 1
6	0,100	0110	0 0 1
7	0,040	0111	1 1 1 0 0
8	0,070	1000	1 0 1 1
9	0,040	1001	1 1 1 0 1
10	0,070	1010	1 1 0 0
11	0,060	1011	0 1 0 1
12	0,060	1100	1 0 0 0
13	0,040	1101	1 1 0 1 0
14	0,040	1110	1 1 0 1 1
15	0,060	1111	1 0 0 1

Pada Tabel 4.1 dan Tabel 4.2 menunjukan terjadi penurunan dan peningkatan jumlah bit yang dihasilkan. Perbandingan hasil bit *Huffman Coding* antara Tabel 4.1 dan Tabel 4.2 dapat dilihat bahwa pada Tabel 4.1 menunjukan nilai probababilitas yang *significant* dapat menghasilkan bit *Huffman Coding* lebih bervariasi antara 2 bit sampai 8 bit, hal ini dikarenakan perbedaaan probabilitas yang tidak merata, berbeda jika dibandingkan dengan Tabel 4.2 yang memiliki nilai probabilitas yang merata menghasilkan bit *Huffman Coding* antara 3 bit sampai 5 bit, hal ini dikarenakan perbedaaan probabilitas yang merata. Terjadinya peningkatan dan penurunan dalam jumlah bit kode *Huffman* tergantung dari probabilitas nilai integer yang dihasilkan. Untuk simbol dengan kekerapan lebih besar memiliki kode *Huffman* yang lebih pendek daripada untuk simbol dengan kekerapan lebih kecil.

Untuk membuktikan bit hasil *Hufman Coding* yang dihasilkan pada Tabel 4.1 adalah benar, maka dilakukan pembuktian dengan pembentukan pohon *Huffman* secara manual dengan melihat nilai probabilitas yang dihasilkan berdasarkan nilai integernya. Berikut ini adalah pembentukan pohon *Huffman* sesuai dengan Tabel 4.1 yang ditunjukan pada Gambar 4.2.



Gambar 4.2 Pembentukan Pohon *Huffman* sesuai Tabel 4.1 secara Manual

Gambar 4.2 menunjukan pembentukan pohon *Huffman* yang disesuaikan dengan Tabel 4.1 secara manual, dapat dibuktikan bahwa bit *Huffman Coding* yang dihasilkan secara program dan secara manual adalah sama. Pada Tabel 4.1 dengan nilai integer 0 menghasilkan bit kode *Huffman*-nya adalah 10 (bit), dan jika dibandingkan dengan Gambar 4.1 menunjukan bahwa nilai integer 0 menghasilkan bit kode *Huffman*-nya adalah 10 (bit). Hal ini membuktikan bahwa program *Huffman Coding* bekerja dengan baik.

Untuk melihat presentasi nilai kompresi *Huffman Coding*, maka dalam pengujian *Huffman Coding* dilakukan percobaan beberapa kali dengan jumlah data yang dibangkitkan tetap dan bervariasi dan nilai probabilitas yang *significant* dan merata Berikut ini adalah tabel hasil pengujian kompresi *huffman coding* dengan jumlah data tetap dan bervariasi pada kondisi nilai probabilitas yang *significant* ditunjukan pada Tabel 4.3 dan Tabel 4.4, sedangkan tabel hasil pengujian kompresi *Huffman Coding* dengan jumlah data tetap dan bervariasi pada kondisi nilai probabilitas yang merata ditunjukan pada Tabel 4.5 dan Tabel 4.6

Tabel 4.3 Hasil Pengujian Kompresi *Huffman Coding* dengan Jumlah Data Tetap (Nilai Probabilitas yang *Significant*)

Jumlah data	Kompresi huffman	Dekompresi huffman	Nilai kompresi (%)
100	82	100	82
100	85	100	85
100	83	100	83
100	82	100	82
100	82	100	82
100	83	100	83
100	83	100	83
100	82	100	82
100	84	100	84
100	85	100	85
Rata-rata nilai kompresi			83,1

Tabel 4.4 Hasil Pengujian Kompresi *Huffman Coding* dengan Jumlah Data Bervariasi (Nilai Probabilitas yang *Significant*)

Jumlah data	Kompresi huffman	Dekompresi huffman	Nilai kompresi (%)
100	82	100	82,00
200	164	200	82,00
300	256	300	85,30
400	325	400	81,25
500	424	500	84,80
600	527	600	87,83
700	609	700	87,00
800	679	800	83,75
900	761	900	84,55
1000	856	1000	85,60
2000	1696	2000	84,80
3000	2561	3000	85,36
4000	3409	4000	85,22
5000	4265	5000	85,30
Rata-rata nilai kompresi			84,62

Tabel 4.5 Hasil Pengujian Kompresi *Huffman Coding* dengan Jumlah Data Tetap (Nilai Probabilitas yang Merata)

Jumlah data	Kompresi huffman	Dekompresi huffman	Nilai kompresi (%)
100	99	100	99
100	97	100	97
100	97	100	97
100	98	100	98
100	99	100	99
100	100	100	100
100	98	100	98
100	99	100	99
100	99	100	99
100	98	100	98
Rata-rata nilai kompresi			98,4

Tabel 4.6 Hasil Pengujian Kompresi *Huffman Coding* dengan Jumlah Data bervariasi (Nilai Probabilitas yang merata)

Jumlah data	Kompresi huffman	Dekompresi huffman	Nilai kompresi (%)
100	99	100	99
200	200	200	100
300	299	300	99,66
400	401	400	100
500	501	500	100
600	601	600	100
700	701	700	100
800	801	800	100
900	901	900	100
1000	1001	1000	100
2000	2001	2000	100
3000	3001	3000	100
4000	4001	4000	100
5000	5001	5000	100
Rata-rata nilai kompresi			100

Pada tabel 4.3 dan 4.4 dalam kondisi nilai probabilitas yang *significant* menunjukan terjadi penurunan jumlah data yang diakibatkan oleh kompresi *Huffman coding* dalam 10 kali pengujian dengan jumlah data yang tetap (100 data), nilai rata-rata hasil kompresi *Huffman Coding* adalah 83,1 % dengan nilai rentang frekuensi ouputnya adalah 82 % sampai 85 %. Sedangkan dalam pengujian yang dilakukan dengan jumlah data yang bervariasi, menunjukan bahwa rata-rata hasil kompresi yang dilakukan oleh *Hufman Coding* adalah 84,62 % dengan nilai rentang frekuensi outputnya antara 81 % sampai 87 %. Hal ini terbukti bahwa program yang dibuat berdasarkan algoritma *Huffman coding* dapat melakukan kompresi data (*encoding*), sehingga data yang akan diproses menjadi lebih kecil dari data originalnya, dan juga program ini dapat melakukan pengembalian data atau penguraian data kembali (*decoding*).

Jika dibandingkan dengan Tabel 4.5 dan 4.6 dalam kondisi nilai probabilitas yang merata menunjukan bahwa dengan jumlah data yang tetap (100 data), nilai rata-rata kompresinya adalah 98,4 % dengan rentang frekuensi ouputnya adalah 97 % sampai 100 %. Sedangkan dalam pengujian yang dilakukan dengan jumlah

data yang bervariasi, menunjukan bahwa rata-rata hasil kompresi yang dilakukan oleh *Hufman Coding* adalah 100 % dengan nilai frekuensi outputnya antara 99 % sampai 100 %. Hal ini menunjukan bahwa *Huffman coding* tidak bekerja dengan baik pada kondisi nilai probabilitas yang merata.

Pada prinsipnya, algoritma *Huffman Coding* ini memanfaatkan nilai probabilitas dari data-data yang akan diproses. Nilai probabilitas dari suatu data yang akan dikompres menggunakan kompresi *Huffman Coding* sangat menentukan dalam pembuatan pohon huffman sebagai pengganti bit kode huffman dari bit kode original. Kode biner untuk simbol dengan kekerapan lebih besar akan memiliki kode yang lebih pendek daripada untuk simbol dengan kekerapan lebih kecil. Untuk itu dibutuhkan perbedaan probabilitas secara *significant* antara data yang satu dengan yang lain.

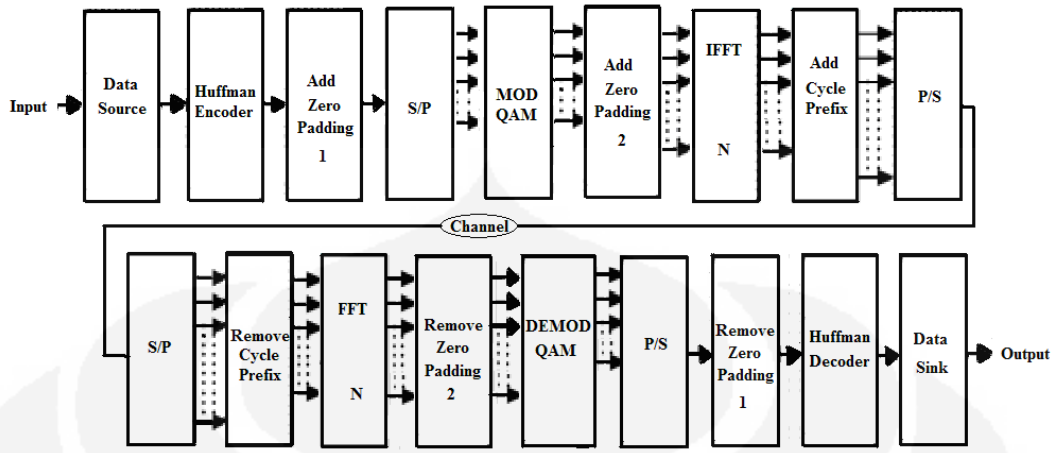
4.2 Pengujian dan Analisa Program *Huffman Coding* dengan OFDM

Dalam menganalisa *Huffman Coding* pada sistem OFDM, akan dianalisa perbandingan nilai input dan output sistem OFDM setelah pengkombinasian dengan *Huffman Coding*, dan juga parameter PAPR pada sistem OFDM.

Parameter-parameter yang digunakan dalam sistem OFDM yang dikombinasikan dengan *Huffman Coding* adalah sebagai berikut:

1. Jumlah data input = 6720
2. Modulasi yang digunakan 16 QAM
3. Jumlah *subcarrier* sebesar 192
4. Panjang IFFT atau FFT sebesar 256
5. *Cycle prefix* sebesar 25 %
6. Kanal yang digunakan tidak dipengaruhi oleh AWGN.

Dalam pengujian nilai input dan output pada sistem OFDM yang dikombinasikan dengan *Huffman Coding* dilakukan dengan 10 kali percobaan. Untuk mengetahui nilai kesalahan (*error*) dapat menggunakan fungsi *bitter* pada Matlab dengan menghitung bit kesalahan antara data input dan output. Berikut ini adalah Gambar 4.3 yang menunjukan sistem pengujian OFDM yang dikombinasikan dengan Huffman Coding, dan hasilnya ditunjukan pada Tabel 4.7.



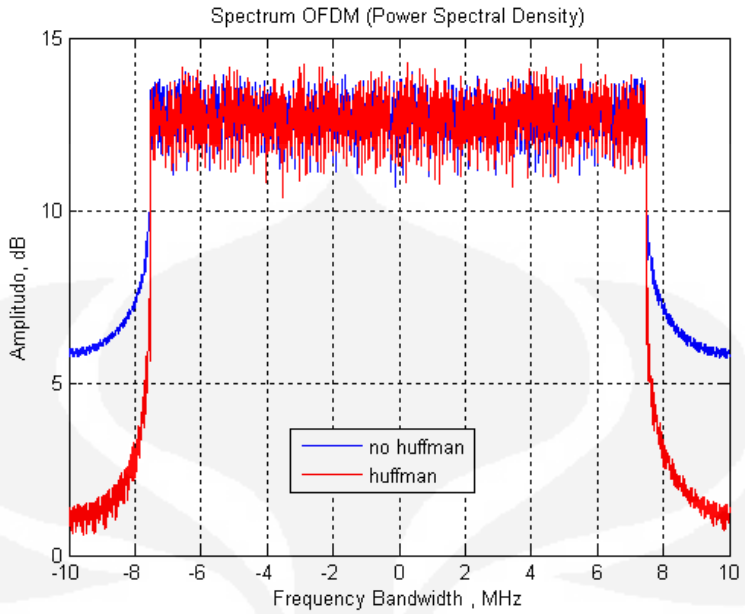
Gambar 4.3 Sistem Pengujian OFDM yang Dikombinasikan dengan *Huffman Coding*

Tabel 4.7 Hasil Pengujian Input dan Output pada Sistem OFDM yang Dikombinasikan dengan *Huffman Coding*

Pengujian ke	Jumlah data	Input OFDM	Output OFDM	Nilai kesalahan
1	6720	6720	6720	0
2	6720	6720	6720	0
3	6720	6720	6720	0
4	6720	6720	6720	0
5	6720	6720	6720	0
6	6720	6720	6720	0
7	6720	6720	6720	0
8	6720	6720	6720	0
9	6720	6720	6720	0
10	6720	6720	6720	0

Berdasarkan Tabel 4.7 dapat dibuktikan bahwa program OFDM yang dikombinasikan dengan *Huffman Coding* bekerja dengan baik. Hal ini dapat dilihat dari nilai input dan output yang dihasilkan pada Tabel 4.7 yang memperlihatkan jumlah data, nilai input dan output OFDM menghasilkan nilai yang sama.

Berikut ini adalah Gambar 4.4 yang menunjukan gambar Spectrum OFDM tanpa dan dengan *Huffman Coding* pada parameter PSD (*Power Spectral Density*).



Gambar 4.4 Spectrum OFDM dengan dan tanpa *Huffman Coding*

Pada Gambar 4.4 menunjukan bahwa terjadi perbedaan nilai PAPR diantara sistem OFDM dengan menggunakan *Huffman Coding* dan tanpa menggunakan *Huffman Coding.* Pada sistem OFDM tanpa *Huffman Coding*, nilai PAPRnya sebesar 6 dB, sedangkan pada sistem OFDM dengan *Huffman Coding*, memiliki nilai PAPR sebesar 1 dB. Terjadi penurunan sebesar 5 dB antara sistem OFDM dengan *Huffman Coding* dan tanpa *Huffman Coding*. Hal ini membuktikan bahwa pengaruh *Huffman Coding* untuk menurunkan nilai PAPR bekerja dengan baik. Bentuk sinyal spectrum OFDM dengan perbedaan nilai *subcarrier* dapat dilihat pada lampiran gambar spectrum OFDM dengan perbedaan nilai *subcarrier*.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisa data hasil simulasi *Huffman Coding* pada sistem OFDM dapat disimpulkan bahwa:

1. *Huffman Coding* adalah salah satu metode kompresi yang baik untuk mengkompresi data sehingga menghasilkan data bit yang lebih kecil dari bit originalnya
2. *Huffman Coding* akan bekerja dengan baik jika data yang akan dikompresi memiliki probabilitas nilai perbedaan yang *significant*, dan tidak bekerja dengan baik jika data yang akan dikompresi memiliki probabilitas nilai perbedaan yang merata.
3. Berdasarkan hasil percobaan dengan nilai probabilitas yang *significant*, pada kondisi jumlah data tetap (100 data), metode *Huffman Coding* mampu mengkompres data dengan rata-rata kompresinya sebesar 83,1 % dari data originalnya, sehingga hasil kompresinya sebesar 16,9 %, dan pada kondisi jumlah data bervariasi (100 data sampai 5000 data), metode *Huffman Coding* mampu mengkompres data dengan rata-rata 84,62 % dari data originalnya, sehingga hasil kompresinya sebesar 15,38 %.
4. Berdasarkan hasil percobaan dengan nilai probabilitas yang merata, pada kondisi jumlah data tetap (100 data), metode *Huffman Coding* hanya mampu mengkompres data dengan rata-rata kompresinya sebesar 98,4 % dari data originalnya, sehingga hasil kompresinya sebesar 1,6 %, dan pada kondisi jumlah data bervariasi (100 data sampai 5000 data), metode *Huffman Coding* tidak dapat melakukan kompresi dengan baik.
5. *Huffman Coding* terbukti mampu menurunkan nilai PAPR pada sistem OFDM sebesar 5 dB, dengan nilai PAPR pada kondisi tanpa *Huffman Coding* sebesar 6 dB, sedangkan nilai PAPR pada kondisi dengan *Huffman Coding* sebesar 1 dB.

6. Penambahan *zero padding* dapat memperbaiki sinyal spectrum OFDM untuk analisa PAPR dengan parameter PSD.
7. Perbedaan nilai *subcarrier* akan mempengaruhi lebar *bandwidth* yang dihasilkan, makin kecil nilai *subcarrier* maka lebar *bandwidth* semakin kecil, begitu pula sebaliknya makin besar nilai *subcarrier* maka lebar *bandwidth* semakin besar.

5.2 Saran

Untuk lebih menurunkan nilai PAPR pada sistem OFDM, maka sistem OFDM dengan *Huffman Coding* ini dapat dikombinasikan dengan menggunakan metode yang sudah pernah dikembangkan sebelumnya, seperti *Selective Mapping* (SLM), *Partial Transmit Sequence* (PTS), *Coding* (*Golay Complementary codes*) dan *Dummy Sequence Insertion* (DSI) sehingga reduksi PAPR pada sistem OFDM dapat lebih maksimal.

DAFTAR ACUAN

[1] Adityo, Wisnu. *Penggunaan Kode Huffman dan Kode Aritmatik pada Entropy Coding*. Program Studi Teknik Informatika ITB, Jalan Ganesha no 10 Bandung

[2] Ayuningtyas, Nadhira. *Implementasi Kode Huffman dalam Aplikasi Kompresi Teks pada Layanan SMS.* Jurusan Teknik Informatika ITB, Jalan Ganesha 10, Bandung

[3] http://en.wikipedia.org/wiki/Orthogonal_Frequency_Division_Multiplexing

[4] http://en.wikipedia.org/wiki/Huffman coding

[5] Kurniawan, Irwan. *Penyandian (Encoding) dan Penguraian Sandi (Decoding) Menggunakan Huffman Coding.* Program Studi Teknik Informatika, Institut Teknologi Bandung Jl. Ganesha 10, Bandung

[6] Langton, Charan. *Orthogonal Frequency Division Multiplexing Tutorial*. www.complextoreal.com 2004

[7] Luis, Anibal Intini. *Orthogonal Frequency Division Multiplexing for Wireless Network.* University of California, Santa Barbara. 2000

[8] Puspito, Sigit W.J. *Mengenal Teknologi Orthogonal Frequency Division Multiplexing (OFDM) pada Komunikasi Wireless.* Elektro Indonesia, Nomor 24, Tahun V, Januari 1999

[9] Xiong, Fuqin. *Digital Modulation Techniques*. ARTECH HOUSE, INC, 2000

[10] Sujaini, Herry, Yessi Mulyani. *Algoritma RUN-LENGTH, HALF-BYTE, & HUFFMAN untuk pemanpatan file.* Program Studi Teknik Informatika, Institut Teknologi Bandung,2000

DAFTAR PUSTAKA

Eltholth, Ashraf A.,et.al. *Peak-to-Average Power Ratio Reduction in OFDM System using Huffman Coding*. Proceedings of world Academy of science, Engineering and Technology Volume 33,pp. 266-270, September 2008.

Harisvan, Joy Tuah Saragih. *Analisis dan Simulasi Kinerja Alokasi Subcarrier pada Orthogonal Frequency Division Multiple Access (OFDMA)*. Skripsi, Sekolah Teknik Elektro dan Informatika,Institute Teknologi Bandung 2007.

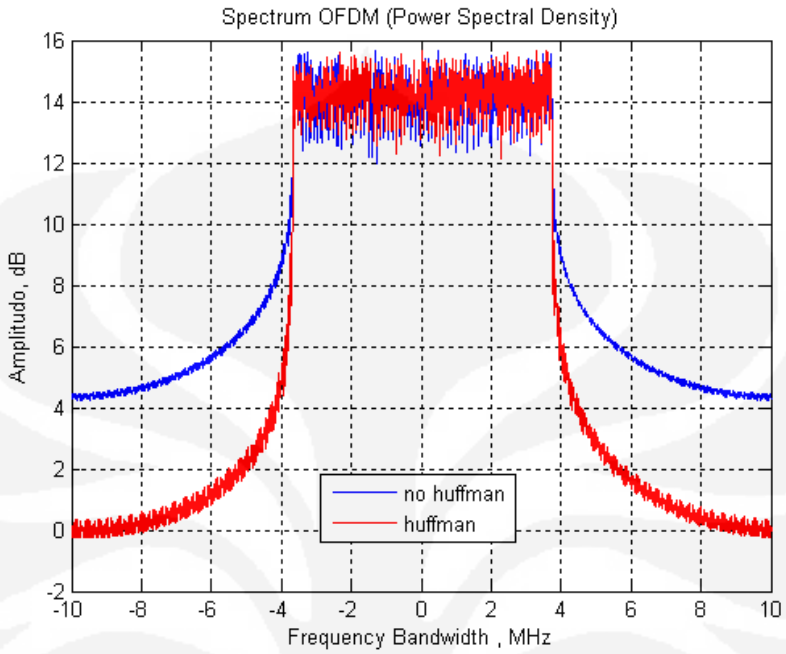
Huang, Kevin, *Reducing The Peak-to-Average Power Ratio in OFDM*, SUID, 2003

Litwin, Louis, Michael Pugel. *The principles of OFDM*. www.rfdesign.com, January 2001

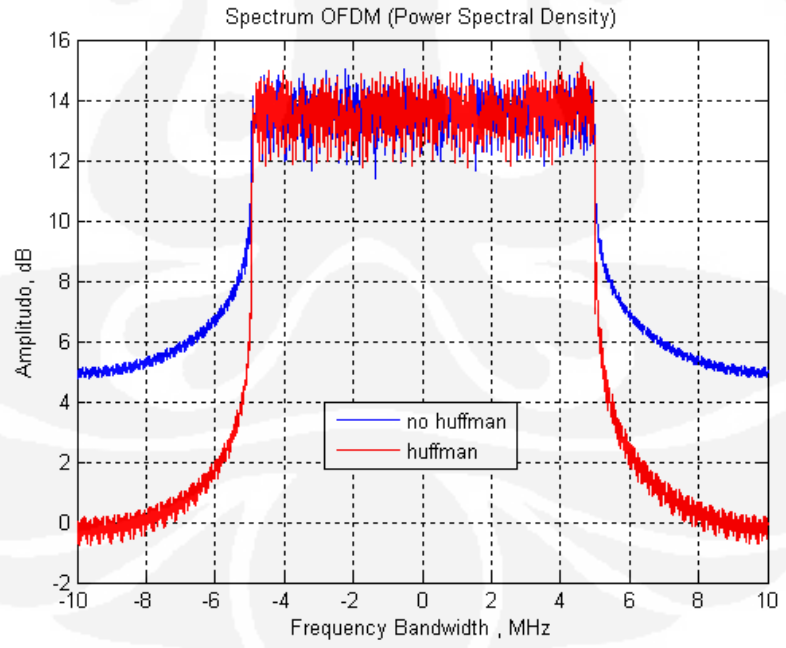
Nurrohmah, Ratnasari. *Teknik Kompresi Lossless Citra Dijital dengan Pengkodean Huffman*. Teknik Elektro Universitas Muhammadiyah Surakarta

Xiong, Haitao, Ping Wang, Zhiyong Bu. *An Efficient Peak-to-Average Power Ratio Reduction Algorithm for WiMAX System.* Institute of Microsystem And Information Technology Chinese Academy of Sciences Shanghai 200050, P. R. China

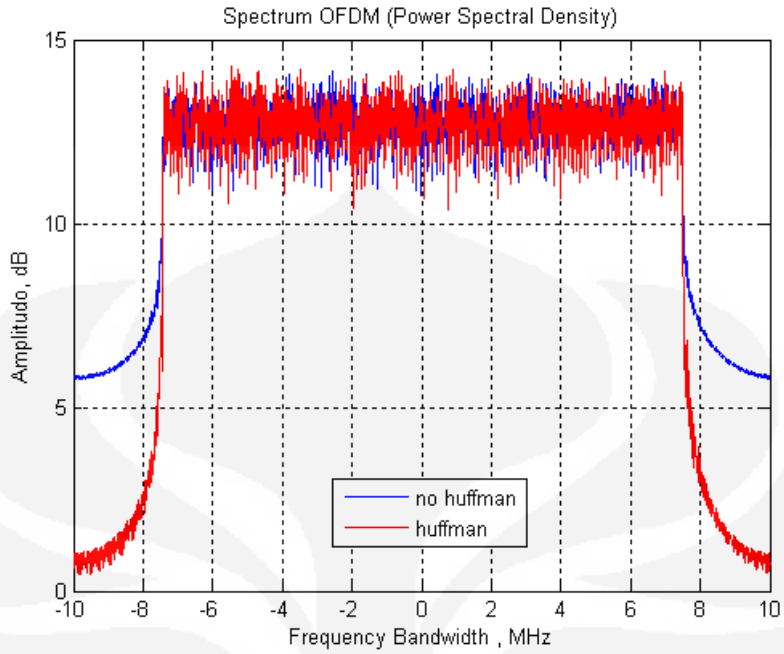
Lampiran gambar Spectrum OFDM dengan perbedaan *Subcarrier*



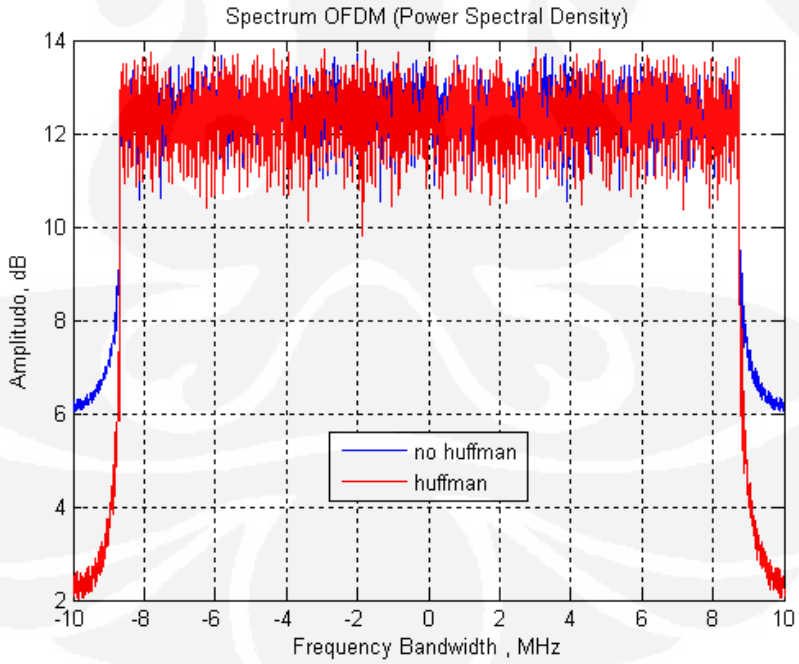
Gambar 1 Spectrum OFDM dengan *Subcarrier* 96



Gambar 2 Spectrum OFDM dengan *Subcarrier* 128

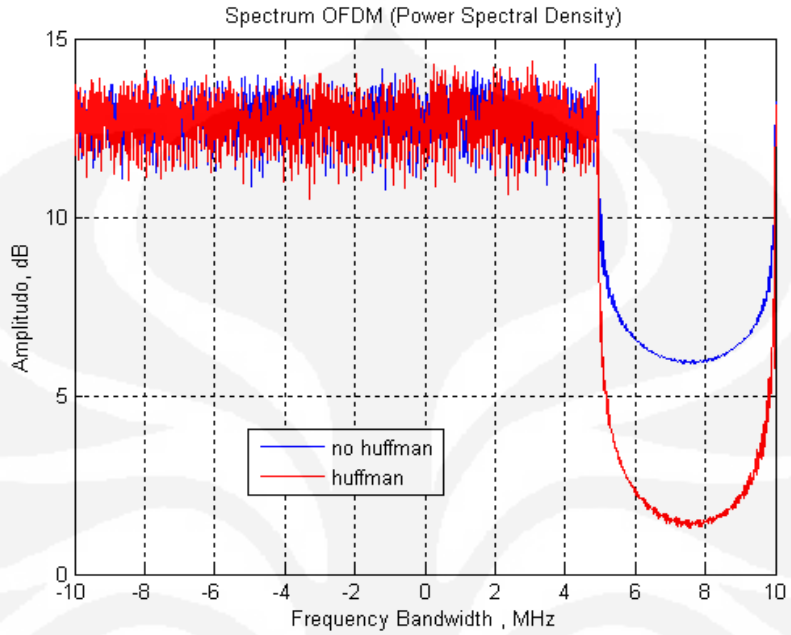


Gambar 3 Spectrum OFDM dengan *Subcarrier* 192

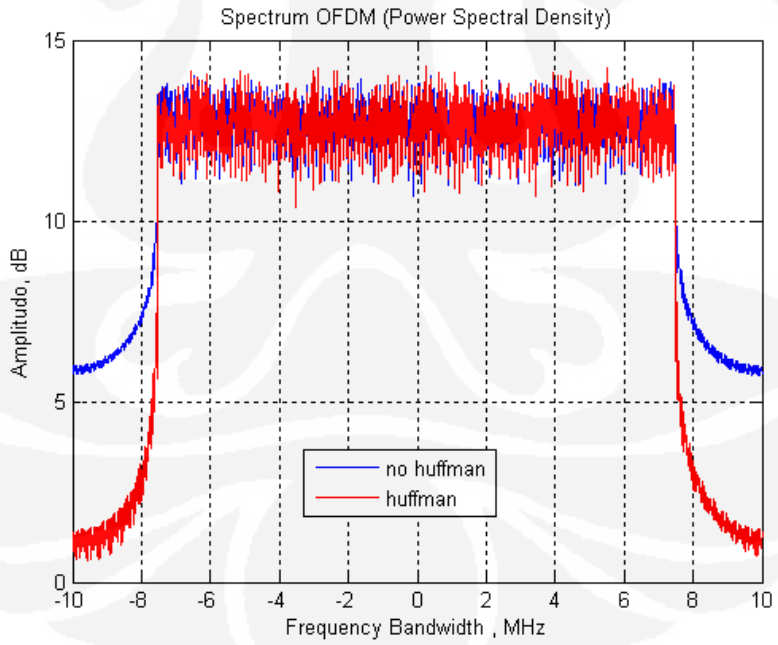


Gambar 4 Spectrum OFDM dengan *Subcarrier* 224

Lampiran gambar Spectrum OFDM dengan penambahan *zero padding* dan tanpa penambahan *zero padding*



Gambar 1 Spectrum OFDM dengan tidak ada penambahan *zero padding* 2



Gambar 2 Spectrum OFDM dengan penambahan *zero padding* 2

Lampiran Data Random yang dibangkitkan sebanyak 1000 data

Data Random yang dibangkitkan sebanyak 1000 data									
Data ke-	**Isi data**	**Data ke-**	**Isi data**	**Data ke-**	**Isi data**	**Data ke-**	**Isi data**	**Data ke-**	**Isi data**
1	0	101	9	201	0	301	0	401	14
2	1	102	0	202	6	302	1	402	3
3	0	103	0	203	7	303	8	403	1
4	1	104	0	204	4	304	0	404	0
5	10	105	8	205	8	305	5	405	1
6	1	106	0	206	1	306	0	406	4
7	0	107	10	207	0	307	5	407	0
8	1	108	2	208	1	308	6	408	2
9	6	109	13	209	5	309	6	409	2
10	4	110	0	210	0	310	4	410	0
11	12	111	0	211	8	311	15	411	7
12	9	112	1	212	2	312	0	412	0
13	2	113	1	213	6	313	0	413	0
14	4	114	1	214	15	314	8	414	7
15	0	115	1	215	1	315	0	415	0
16	2	116	1	216	6	316	2	416	0
17	1	117	11	217	4	317	0	417	1
18	10	118	5	218	5	318	3	418	2
19	9	119	13	219	0	319	0	419	2
20	2	120	0	220	4	320	0	420	4
21	7	121	6	221	13	321	2	421	4
22	4	122	1	222	3	322	10	422	15
23	0	123	1	223	7	323	3	423	1
24	0	124	0	224	0	324	0	424	0
25	2	125	3	225	8	325	13	425	5
26	0	126	1	226	4	326	0	426	2
27	0	127	7	227	0	327	8	427	0
28	1	128	8	228	2	328	3	428	3
29	11	129	1	229	0	329	9	429	8
30	0	130	0	230	2	330	0	430	13
31	0	131	1	231	0	331	4	431	1
32	1	132	14	232	3	332	0	432	1
33	9	133	2	233	1	333	2	433	15
34	0	134	1	234	0	334	5	434	3
35	2	135	7	235	0	335	0	435	13
36	1	136	1	236	0	336	1	436	11
37	1	137	11	237	0	337	0	437	8
38	1	138	0	238	13	338	3	438	5

Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data
39	10	139	1	239	14	339	3	439	0
40	7	140	12	240	3	340	0	440	3
41	10	141	2	241	14	341	0	441	4
42	1	142	0	242	5	342	0	442	7
43	8	143	6	243	0	343	1	443	0
44	0	144	0	244	8	344	7	444	0
45	0	145	3	245	12	345	3	445	9
46	1	146	13	246	0	346	1	446	0
47	1	147	1	247	5	347	0	447	0
48	2	148	6	248	0	348	2	448	0
49	14	149	5	249	4	349	5	449	8
50	1	150	14	250	1	350	10	450	0
51	0	151	0	251	12	351	6	451	3
52	3	152	3	252	0	352	15	452	12
53	2	153	13	253	4	353	10	453	1
54	0	154	0	254	1	354	2	454	0
55	2	155	2	255	2	355	11	455	1
56	2	156	4	256	2	356	0	456	14
57	9	157	1	257	5	357	4	457	6
58	8	158	2	258	0	358	13	458	4
59	9	159	6	259	6	359	0	459	7
60	3	160	0	260	10	360	0	460	2
61	3	161	0	261	1	361	5	461	0
62	3	162	1	262	1	362	3	462	0
63	0	163	5	263	13	363	0	463	7
64	0	164	12	264	10	364	10	464	4
65	1	165	2	265	5	365	2	465	4
66	3	166	0	266	0	366	11	466	2
67	0	167	2	267	9	367	7	467	12
68	7	168	13	268	14	368	4	468	7
69	1	169	1	269	0	369	14	469	1
70	0	170	1	270	15	370	0	470	6
71	3	171	7	271	1	371	0	471	1
72	3	172	0	272	0	372	6	472	3
73	0	173	0	273	0	373	0	473	0
74	0	174	5	274	5	374	1	474	0
75	12	175	0	275	2	375	1	475	4
76	0	176	1	276	0	376	0	476	11
77	0	177	9	277	0	377	1	477	13
78	0	178	10	278	14	378	0	478	0

Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data
79	3	179	5	279	1	379	15	479	0
80	2	180	6	280	0	380	11	480	1
81	0	181	14	281	0	381	1	481	0
82	4	182	1	282	4	382	0	482	0
83	1	183	0	283	6	383	0	483	5
84	0	184	6	284	0	384	0	484	14
85	1	185	11	285	0	385	11	485	1
86	0	186	0	286	11	386	1	486	8
87	12	187	0	287	7	387	3	487	0
88	4	188	2	288	3	388	0	488	9
89	10	189	0	289	5	389	6	489	0
90	4	190	4	290	14	390	0	490	0
91	8	191	13	291	3	391	1	491	1
92	1	192	1	292	7	392	10	492	3
93	14	193	2	293	3	393	9	493	0
94	0	194	0	294	14	394	8	494	10
95	2	195	2	295	0	395	0	495	1
96	2	196	1	296	14	396	11	496	2
97	0	197	0	297	0	397	3	497	1
98	1	198	0	298	11	398	0	498	0
99	0	199	6	299	4	399	0	499	14
100	2	200	0	300	1	400	10	500	14

Data Random yang dibangkitkan sebanyak 1000 data

Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data
501	13	601	10	701	5	801	4	901	1
502	1	602	0	702	0	802	4	902	1
503	1	603	12	703	9	803	0	903	0
504	0	604	4	704	11	804	1	904	1
505	0	605	0	705	0	805	5	905	5
506	9	606	4	706	0	806	1	906	2
507	4	607	6	707	1	807	0	907	15
508	1	608	7	708	14	808	3	908	0
509	0	609	8	709	4	809	9	909	3
510	1	610	2	710	5	810	11	910	2
511	6	611	2	711	1	811	10	911	1
512	0	612	1	712	9	812	13	912	2
513	0	613	4	713	8	813	7	913	14
514	1	614	0	714	2	814	1	914	8

Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data
515	10	615	2	715	0	815	0	915	0
516	4	616	0	716	0	816	0	916	9
517	0	617	0	717	3	817	12	917	7
518	1	618	0	718	0	818	2	918	1
519	0	619	8	719	6	819	0	919	6
520	0	620	1	720	2	820	1	920	0
521	0	621	2	721	0	821	9	921	3
522	0	622	12	722	12	822	0	922	7
523	1	623	1	723	0	823	0	923	0
524	5	624	4	724	0	824	8	924	3
525	8	625	0	725	5	825	0	925	0
526	6	626	7	726	1	826	0	926	10
527	0	627	1	727	1	827	12	927	0
528	3	628	7	728	11	828	0	928	0
529	2	629	1	729	13	829	1	929	0
530	12	630	7	730	2	830	2	930	6
531	1	631	0	731	0	831	8	931	2
532	0	632	0	732	8	832	1	932	0
533	13	633	4	733	0	833	8	933	0
534	0	634	0	734	0	834	0	934	6
535	3	635	12	735	5	835	13	935	14
536	12	636	2	736	0	836	9	936	5
537	11	637	0	737	0	837	14	937	8
538	4	638	0	738	0	838	0	938	0
539	5	639	0	739	6	839	2	939	7
540	12	640	0	740	0	840	1	940	10
541	1	641	0	741	5	841	0	941	8
542	0	642	9	742	0	842	0	942	6
543	4	643	6	743	4	843	5	943	11
544	2	644	5	744	2	844	13	944	1
545	1	645	1	745	3	845	0	945	4
546	7	646	2	746	0	846	8	946	0
547	9	647	14	747	11	847	0	947	14
548	0	648	7	748	0	848	15	948	1
549	3	649	1	749	0	849	0	949	1
550	7	650	13	750	8	850	0	950	4
551	0	651	0	751	15	851	2	951	1
552	9	652	2	752	0	852	3	952	9
553	5	653	2	753	14	853	8	953	2
554	8	654	2	754	12	854	0	954	0

Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data
555	12	655	1	755	13	855	0	955	0
556	1	656	2	756	3	856	0	956	3
557	2	657	8	757	13	857	3	957	13
558	0	658	0	758	0	858	0	958	13
559	12	659	1	759	0	859	4	959	6
560	5	660	0	760	0	860	0	960	0
561	4	661	0	761	1	861	2	961	2
562	15	662	8	762	0	862	2	962	0
563	2	663	5	763	1	863	5	963	0
564	1	664	10	764	0	864	0	964	6
565	9	665	0	765	0	865	0	965	7
566	0	666	2	766	0	866	12	966	1
567	3	667	1	767	3	867	0	967	1
568	0	668	1	768	10	868	0	968	0
569	1	669	11	769	0	869	1	969	0
570	0	670	0	770	3	870	4	970	3
571	4	671	1	771	2	871	0	971	9
572	13	672	14	772	0	872	1	972	0
573	0	673	0	773	0	873	5	973	2
574	8	674	0	774	1	874	1	974	0
575	5	675	0	775	1	875	12	975	0
576	14	676	0	776	15	876	7	976	2
577	0	677	0	777	12	877	1	977	5
578	0	678	0	778	1	878	0	978	0
579	6	679	2	779	12	879	1	979	14
580	8	680	7	780	2	880	0	980	2
581	0	681	1	781	14	881	0	981	5
582	5	682	9	782	13	882	0	982	8
583	0	683	8	783	6	883	0	983	2
584	0	684	0	784	4	884	10	984	2
585	10	685	4	785	10	885	1	985	6
586	0	686	0	786	0	886	7	986	0
587	6	687	0	787	1	887	8	987	13
588	0	688	11	788	0	888	1	988	0
589	1	689	0	789	1	889	8	989	2
590	3	690	0	790	0	890	1	990	11
591	4	691	9	791	8	891	0	991	10
592	2	692	5	792	0	892	2	992	4
593	0	693	0	793	1	893	0	993	0
594	5	694	8	794	0	894	3	994	1

Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data	Data ke-	Isi data
595	0	695	1	795	5	895	14	995	1
596	6	696	0	796	1	896	2	996	0
597	1	697	2	797	3	897	8	997	13
598	1	698	12	798	1	898	1	998	2
599	1	699	1	799	0	899	0	999	5
600	0	700	2	800	0	900	4	1000	10