UNIVERSITAS INDONESIA

RANCANG BANGUN RANGKAIAN PENGIRIM OFDM DENGAN *HUFFMAN CODE* PADA DSK TMS320C6713 MENGGUNAKAN SIMULINK

SKRIPSI

NOVAN FERDIAN DJAFAR
04 05 03 060 5

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
DESEMBER 2009

UNIVERSITAS INDONESIA

RANCANG BANGUN RANGKAIAN PENGIRIM OFDM DENGAN *HUFFMAN CODE* PADA DSK TMS320C6713 MENGGUNAKAN SIMULINK

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

NOVAN FERDIAN DJAFAR
04 05 03 060 5

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK

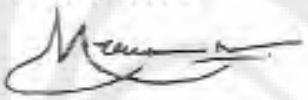
DESEMBER 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri.

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	:	Novan Ferdian Djafar
NPM	:	04 05 03 060 5
Tanda Tangan	:	
Tanggal	:	28 Desember 2009

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama	:	Novan Ferdian Djafar
NPM	:	0405030605
Program Studi	:	Teknik Elektro
Judul Skripsi	:	**RANCANG BANGUN RANGKAIAN PENGIRIM OFDM DENGAN *HUFFMAN CODE* PADA DSK TMS320C6713 MENGGUNAKAN SIMULINK**

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

DEWAN PENGUJI

Pembimbing : Dr. Ir. Arman Djohan Diponegoro, M.Eng ( )

Penguji : Prof. Dr. Ir. Dadang Gunawan, M.Eng ( )

Penguji : Filbert Hilman Juwono S.T., M.T. ( )

Ditetapkan di : Depok

Tanggal :28 Desember 2009

UCAPAN TERIMA KASIH

Puji syukur yang sebesar-besarnya saya sampaikan ke hadirat Allah Subhanahu Wa Ta'ala, karena atas segala karunia dan rahmat-Nya saya mampu menyelesaikan Skripsi ini. Saya menyadari bahwa dalam penggarapan Skripsi ini saya telah terbantu oleh berbagai pihak. Maka dari itu saya mengucapkan terima kasih kepada:

(1) Dr. Ir. Arman Djohan Diponegoro, M.Eng selaku dosen pembimbing saya yang telah bersedia meluangkan waktunya untuk memberikan bimbingan, petunjuk dan saran, serta berkontribusi dalam menentukan judul karya Skripsi ini sehingga karya Skripsi ini dapat diselesaikan dengan baik.

(2) Dr. Abdul Muis, ST., M.Eng selaku koordinator Skripsi, Departemen Teknik Elektro, Fakultas Teknik Universitas Indonesia.

(3) Seluruh keluarga besar sivitas akademika Fakultas Teknik, Universitas Indonesia, khususnya karyawan sekretariat Departemen Teknik Elektro yang telah banyak memberikan bantuan dalam urusan administrasi Skripsi.

(4) Orang tua serta keluarga saya yang telah memberikan bantuan material maupun dukungan moral yang sangat besar dalam penyusunan Skripsi ini.

(5) Dan juga teman-teman saya, baik yang terlibat langsung ataupun tidak langsung dalam menolong saya menyelesaikan Skripsi ini.

Akhir kata, saya harapkan Allah Subhanahu Wa Ta'ala berkenan membalas kebaikan semua pihak yang telah membantu dalam penggarapan Skripsi ini. Semoga Skripsi ini membawa manfaat bagi pengembangan teknologi dan ilmu pengetahuan

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

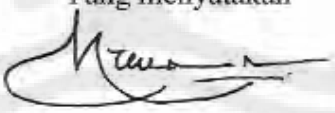
Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Novan Ferdian Djafar
NPM	: 04 05 03 060 5
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 :
Rancang Bangun Rangkaian Pengirim OFDM dengan *Huffman Code* pada DSK TMS320C6713 Menggunakan Simulink
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 memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 28 Desember 2009

Yang menyatakan



(Novan Ferdian Djafar)

ABSTRAK

Nama : Novan Ferdian Djafar
Program Studi : Teknik Elektro

Judul : **Rancang Bangun Rangkaian Pengirim OFDM dengan *Huffman Code* pada DSK TMS320C6713 Menggunakan Simulink**

Dalam bidang komunikasi, ketersediaan *bandwidth* yang semakin terbatas mendorong timbulnya kebutuhan akan teknologi-teknologi yang dapat memanfaatkan *bandwidth* yang tersedia seoptimal mungkin. Salah satu teknologi yang memungkinkan penggunaan *bandwidth* lebih maksimal ialah *Orthogonal Frequency Division Multiplexing* (OFDM). OFDM merupakan salah satu teknik transmisi yang menggunakan beberapa buah *frequency subcarrier* yang saling tegak lurus (*orthogonal*). Karakteristik yang saling *orthogonal* membuat *frequency* sub*carrier* dapat saling *overlap* tanpa menimbulkan Interferensi.

Huffman coding adalah suatu metode kompresi data dengan cara pembentukan pohon *Huffman* melalui proses *encoding* menyebabkan data tersebut dapat dikompresi sehingga penggunaan *Huffman coding* pada rangkaian pengirim OFDM dapat semakin mengoptimalkan penggunaan *bandwidth*. Pada Skripsi ini, dilakukan rancang bangun rangkaian pengirim OFDM dengan *Huffman coding* pada DSK (*Digital Signal Processing Starter Kit*) TMS320C6713 menggunakan Simulink. Dari hasil rancang bangun didapatkan bahwa rangkaian pengirim OFDM dengan *Huffman code* dapat dibangun dengan menggunakan Prosesor DSP (*Digital Signal Processing*).

Kata kunci:
OFDM*, Huffman Coding,* Ortogonal*,* IFFT, QAM, Simulink, Prosesor DSP.

ABSTRACT

Name : Novan Ferdian Djafar
Study Program : *Electrical Engineering*
Title : ***Design and Construction of* OFDM *Transmitter with Huffman Code on* DSK TMS320C6713 *Using* Simulink**

In the communication field, bandwidth availability that is becoming more limited has caused the need for technologies that can make use of the bandwidth available optimally. One of the technologies that could maximize the use of bandwidth is Orthogonal Frequency Division Multiplexing (OFDM) OFDM *is one of the transmission techniques that uses frequency subcarriers which is orthogonal to each other. The frequency subcarriers orthogonality enables them to overlap with each other without producing interference.*

Huffman coding is a method of data compression by making a Huffman tree in the encoding process which can causes data to be compress so that the use of Huffman coding in the OFDM *transmitter can further optimize the bandwidth usage. In this research, the design and development of* OFDM *Transmitter with Huffman Coding is built into* DSK *(Digital Signal Processing Starter Kit*) TMS 320C6713 *using Simulink. The result shows that* OFDM *Transmitter with Huffman Coding can be built by using* DSP *(Digital Signal Processing) Processor.*

Key words:
OFDM*, Huffman Coding, Orthogonal,* IFFT, QAM, Simulink*,* DSP *Processor.*

DAFTAR ISI

Halaman

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

O*rthogonal frequency division multiplexing* (OFDM) merupakan salah satu teknik transmisi yang memungkinkan spektrum sinyalnya dibagi-bagi ke dalam berbagai *sub-carier* yang saling berpotongan tanpa mengalami interferensi satu dengan lain. Teknologi ini juga membuat informasi yang dipancarkan lebih tahan terhadap *frequency selective fading*. Keunggulan tersebut yang membedakan teknik transmisi *frequency multicarrier* OFDM dengan teknik transmisi *frequency multicarrier* yang telah ada sebelumnya (*Frequency Division Multiplexing*). Huffman Coding merupakan metode kompresi data yang dapat meminimumkan ukuran data. Metode kompresi Huffman Coding diharapkan dapat diimplementasikan pada sistem OFDM.

Pada Skripsi ini dilakukan proses rancang bangun rangkaian pengirim OFDM dengan Huffman Coding dengan menggunakan Simulink dan diterapkan pada *Digital Signal Processing Starter Kit* (DSK) TMS320C6713. Pembuatan dari rangkaian pengirim OFDM dengan Huffman Coding ini dibuat dengan papan DSK TMS320C6713 dengan terlebih dahulu membuat model simulasi pemancar OFDM dengan Huffman Coding pada Simulink. Setelah model pada Simulink selesai dibuat, maka model telah siap untuk diterapkan pada DSK. Penerapan model pada DSK tidak membutuhkan pemrograman ulang karena Simulink akan membuatkan kode bahasa pemrograman C untuk dapat diterapkan dalam DSK. Kode bahasa pemrograman ini akan dijalankan dalam perangkat lunak Code Composer Studio (CCS) dan dari perangkat lunak ini kemudian program dijalankan pada DSK.

1.2 Perumusan Masalah

Masalah yang dibahas adalah pembuatan rancang bangun rangkaian pengirim OFDM dengan *Huffman Code* pada DSK TMS320C6713. Pembuatan dimulai dengan membuat model pada Simulink untuk kemudian diterapkan pada DSK dengan menggunakan perangkat lunak tambahan *Code*

Composer Studio (CCS). Beberapa penyesuaian harus dilakukan agar blok yang dibuat dapat diimplementasikan pada DSK.

1.3 Tujuan Penelitian

Tujuan penelitian ini adalah untuk mendapatkan rancang bangun rangkaian pengirim OFDM dengan *Huffman Code* pada DSK TMS320C6713 dengan menggunakan Simulink.

1.4 Batasan Masalah

Masalah dibatasi hanya pada simulasi dan rancang bangun rangkaian pengirim OFDM dengan *Huffman encoder* dengan menggunakan MATLAB R2008a dan penerapannya pada DSK TMS320C6713. Hal-hal yang tidak dilakukan pada Skripsi ini adalah :

1. Tidak melakukan uji coba dan tidak membahas tentang rangkaian penerima OFDM dengan *Huffman decoder* .
2. Tidak melakukan uji coba dengan berbagai macam jenis gangguan.

1.5 Metodologi Penelitian

Metodologi yang digunakan adalah merancang rangkaian pengirim OFDM dengan *Huffman Code*. Hal pertama yang dilakukan adalah mempelajari Huffman Coding dan kemudian mempelajari tentang OFDM. Penelitian kemudian dilanjutkan dengan membangun rangkaian pengirim OFDM dengan *Huffman encoder* pada blok model simulasi pada Simulink. Setelah hasil yang dikehendaki didapatkan, maka selanjutnya adalah menerapkan model simulasi pada DSK TMS320C6713. Penerapan pada DSK TMS320C6713 dilakukan dengan bantuan perangkat lunak tambahan yaitu *Code Composer Studio* (CCS).

1.6 Sistematika Penulisan

BAB 1 PENDAHULUAN

Bab ini menjelaskan tentang latar belakang masalah, perumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan tentang dasar teori *Huffman coding*, prinsip dasar algoritma *Huffman coding*, sistem OFDM, sinyal ortogonal, prinsip kerja OFDM, modulasi QAM, transformasi Fourier.

BAB 3 RANCANG BANGUN

Bab ini membahas pembuatan rancang bangun rangkaian pengirim OFDM dengan *Huffman code* baik pada perangkat lunak Simulink serta penerapannya pada perangkat keras DSK TMS320C6713.

BAB 4 UJI COBA DAN ANALISA

Bab ini berisi pembahasan tentang pengujian serta analisa dari model rangkaian pengirim OFDM dengan *Huffman code* yang telah dibuat pada Simulink dan diterapkan pada DSK TMS320C6713.

BAB 5 KESIMPULAN

Memberikan kesimpulan dari Skripsi ini.

BAB 2
LANDASAN TEORI

2.1 *Huffman Coding*

Huffman coding menggunakan tabel kode dengan panjang bervariasi untuk melakukan *encoding* dari sebuah simbol. Tabel kode dengan panjang bervariasi tersebut telah dibuat terlebih dahulu secara terpisah berdasarkan nilai probabilitas 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 construction of minimum-redundancy codes*". Biasanya *Huffman coding* digunakan pada aplikasi seperti kompresi teks, data atau citra dijital.[4]

Huffman coding menggunakan metode spesifik untuk merepresentasikan setiap simbol yang menghasilkan suatu kode prefiks. Kode prefiks ini adalah sekumpulan kode biner yang pada kode ini tidak mungkin terdapat kode prefiks yang menjadi awalan bagi kode biner yang merepresentasikan simbol lain. Hal ini akan mencegah timbulnya keambiguan dalam proses *decoding*. Dalam *Huffman coding*, kode biner untuk simbol dengan kemunculan lebih besar akan memiliki kode yang lebih pendek daripada untuk simbol dengan kemunculan yang lebih sedikit.[2]

Proses *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 kemunculan 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 simpulnya berupa simpul daun yang mengandung simbol itu sendiri serta bobot atau probabilitasnya dari simbol tersebut dan bisa juga mengandung *link* ke simpul orang tua. Kemudian dibuat suatu perjanjian berdasarkan posisi anak simpulnya, contohnya jika bit (*binary digit*) '0' akan merepresentasikan anak kiri maka bit '1' akan merepresentasikan anak kanan dari simpulnya. 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 probabilitas paling kecil dan dikombinasikan sebagai suatu simpul orang tua.
3. Membuat simpul yang baru yang merupakan simpul orang tua dari dua simpul dengan probabilitas paling kecil.
4. Memilih sebuah simpul berikutnya (termasuk simpul baru) yang memiliki peluang terkecil.
5. Melakukan prosedur yang sama pada dua simbol berikutnya yang memiliki probabilitas terkecil.

2.2 Prinsip Dasar Algoritma *Huffman Coding* [8]

Dalam algoritma *Huffman coding* setiap simbol atau karakter misalnya untuk ASCII, yang biasanya diwakili oleh 8 bits, jika ada suatu data yang berisi deretan karakter "ABACAD" maka ukuran data tersebut tanpa *Huffman coding* adalah 6 x 8 bits = 48 bits atau 6 *bytes*. Jika setiap karakter tersebut diberi kode lain, misalnya A=1, B=00, C=010, dan D=011, maka ukurannya menjadi 11 bits (10010101011), yang perlu diperhatikan ialah bahwa kode-kode tersebut harus unik dan kodenya harus tidak dapat dibentuk dari kode-kode yang lain. Misalnya pada contoh di atas jika kode D diganti 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 untuk simbol yang lebih sering muncul kodenya memiliki jumlah bit yang lebih sedikit dibandingkan dengan simbol yang jarang muncul. Seperti pada contoh di atas, karakter A lebih sering muncul (3 kali kemunculan), sehingga kodenya memiliki jumlah bit yang lebih sedikit dibandingkan simbol lainnya.

Dalam pembentukan kode untuk masing-masing simbol terdapat kriteria bahwa kode harus unik dan simbol yang kemunculannya lebih banyak akan memiliki jumlah bit yang lebih sedikit. Sebagai contoh, sebuah data yang berisi karakter-karakter "PERKARA" akan dimampatkan. 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 ke dalam rangkaian bit, "PERKARA" menjadi :

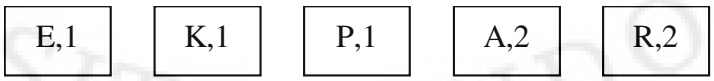
01010000010001010101001001001011010000010101001001000001
P E R K A R A

yang berukuran 56 bit.

Tugas kita yang pertama 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 paling sedikit :

E = 1
K = 1
P = 1
A = 2
R = 2

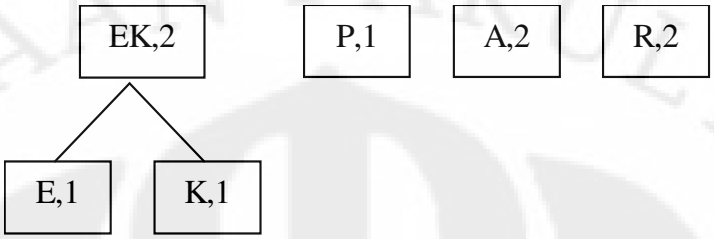
Untuk karakter yang memiliki frekuensi kemunculan sama seperti E, K dan P disusun menurut kode ASCII-nya, begitu pula untuk A dan R.
Selanjutnya buatlah simpul masing-masing karakter beserta frekuensinya sebagai berikut :



Gambar 2.1 Masing-masing karakter beserta frekuensinya

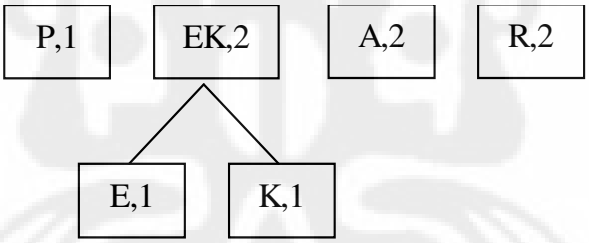
Ambil 2 simpul yang paling kiri (P dan E), lalu buat simpul baru yang merupakan gabungan dua simpul tersebut, simpul gabungan ini akan memiliki

cabang masing-masing 2 simpul yang digabungkan tersebut. Frekuensi dari simpul gabungan ini adalah jumlah frekuensi cabang-cabangnya. Jika kita gambarkan akan menjadi seperti berikut ini :



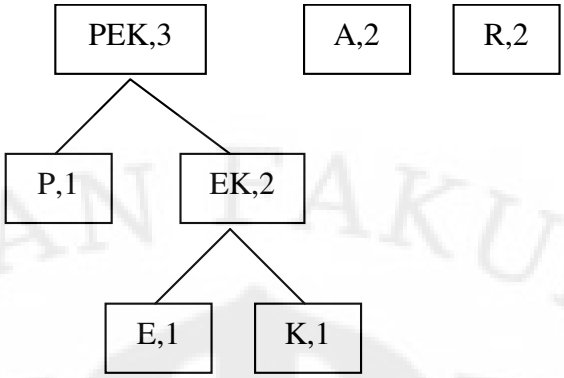
Gambar 2.2 Frekuensi dari simpul gabungan

Jika dilihat frekuensi tiap simpul pada level paling atas adalah EK=2, P=1, A=2, dan R=2. Simpul-simpul tersebut harus diurutkan lagi dari yang paling sedikit jumlahnya, sehingga simpul EK harus digeser ke sebelah kanan simpul P dan jika menggeser suatu simpul yang memiliki cabang, maka seluruh cabangnya harus diikutkan juga. Setelah diurutkan hasilnya akan menjadi sebagai berikut :



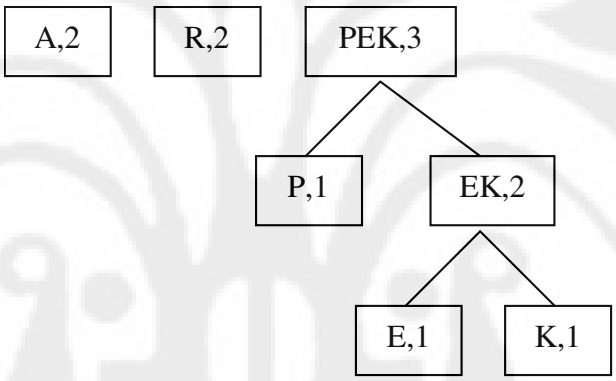
Gambar 2.3 Frekuensi dari simpul gabungan setelah diurutkan kembali

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



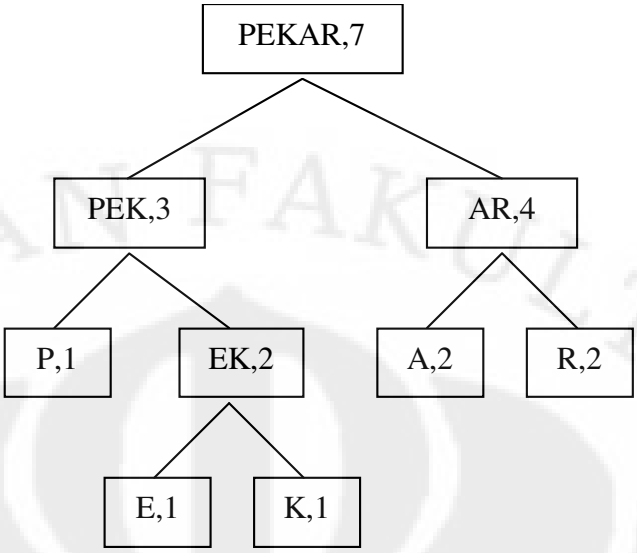
Gambar 2.4 Simpul P digabung dengan simpul EK menjadi simpul PEK

Kemudian diurutkan lagi menjadi :



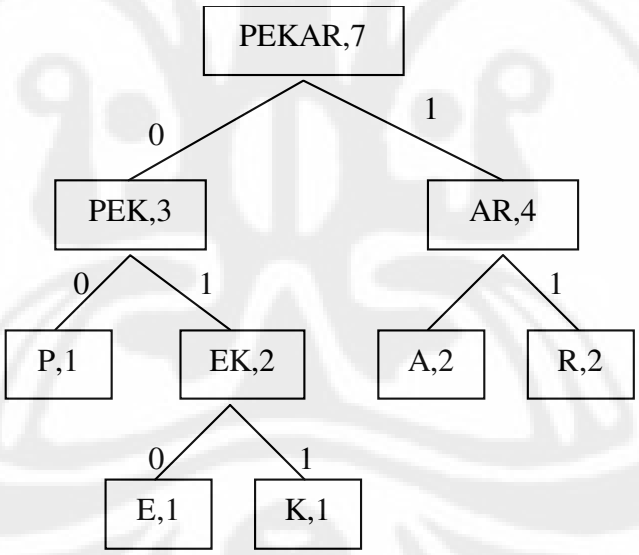
Gambar 2.5 Pengurutan kembali pohon Huffman

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



Gambar 2.6 Pohon Huffman terbentuk

Setelah pohon Huffman terbentuk, berikan tanda bit 0 untuk setiap cabang kiri dan bit 1 untuk setiap cabang kanan seperti gambar berikut :



Gambar 2.7 Pohon Huffman untuk string "PERKARA"

Untuk mendapatkan kode Huffman masing-masing karakter, telusuri karakter tersebut dari simpul yang paling atas (PEKAR) sampai ke simpul karakter tersebut dan susunlah bit-bit yang dilaluinya. Untuk mendapatkan kode Karakter E, dari simpul PEKAR kita harus menuju ke simpul PEK melalui bit 0 dan selanjutnya menuju ke simpul EK melalui bit 1, dilanjutkan ke simpul E

melalui bit 0, jadi kode dari karakter E adalah 010. Untuk mendapatkan kode Karakter K, dari simpul PEKAR kita harus menuju ke simpul PEK melalui bit 0 dan selanjutnya menuju ke simpul EK melalui bit 1, dilanjutkan ke simpul K melalui bit 1, jadi kode dari karakter K adalah 011. Untuk mendapatkan kode Karakter P, dari simpul PEKAR kita harus menuju ke simpul PEK melalui bit 0 dan selanjutnya menuju ke simpul P melalui bit 0, jadi kode dari karakter P adalah 00. Untuk mendapatkan kode Karakter A, dari simpul PEKAR kita harus menuju ke simpul AR melalui bit 1 dan selanjutnya menuju ke simpul A melalui bit 0, jadi kode dari karakter A adalah 10.

Terakhir, untuk mendapatkan kode Karakter R, dari simpul PEKAR kita harus menuju ke simpul AR melalui bit 1 dan selanjutnya menuju ke simpul R melalui bit 1, jadi kode dari karakter R adalah 11.

Hasil akhir kode Huffman dari data di atas adalah :

E = 010
K = 011
P = 00
A = 10
R = 11

Dengan kode ini, data yang berisi karakter-karakter "PERKARA" ukurannya akan menjadi lebih kecil, yaitu :

00 010 11 011 10 11 10 = 16 bit
P E R K A R A

Dengan Algoritma Huffman berarti data ini dapat dihemat sebanyak 56 bit -16 bit = 40 bit.

Untuk proses pengembalian ke data aslinya, kita harus mengacu kembali kepada Tabel kode Huffman yang telah dihasilkan. Untuk contoh yang di atas, hasil pemampatannya adalah :

000101101100 1110

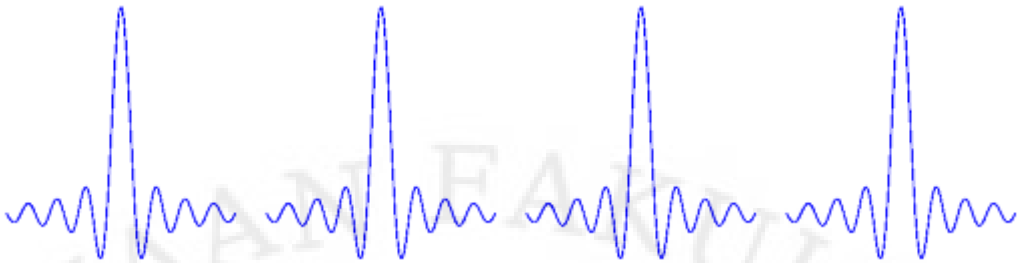
dengan kode Huffman :

E = 010
K = 011
P = 00
A = 10
R = 11

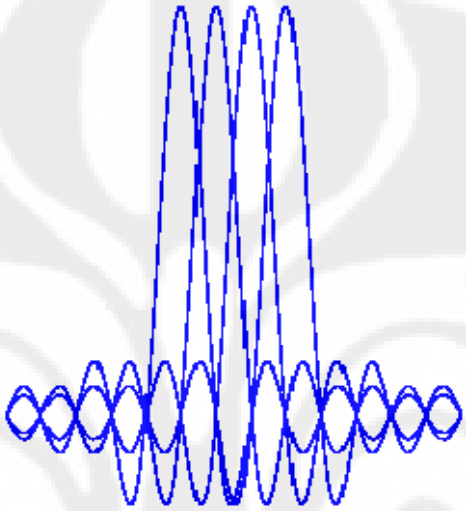
Ambillah satu-persatu bit hasil pemampatan mulai dari kiri, jika bit tersebut termasuk dalam daftar kode, lakukan pengembalian, jika tidak ambil kembali bit selanjutnya dan gabungkan bit tersebut. Bit pertama dari hasil pemampatan di atas adalah 0, karena 0 tidak termasuk dalam daftar kode Huffmannya, jadi 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 maka diambil bit keempat yaitu 1 dan kita gabungkan menjadi 01. 01 juga tidak terdapat dalam daftar, jadi diambil kembali bit selanjutnya yaitu 0 dan digabungkan menjadi 010. 010 terdapat dalam daftar kode yaitu karakter E. Demikian seterusnya sampai bit terakhir sehingga akan didapatkan hasil pengembalian yaitu PERKARA.

2.3 *Orthogonal Frequency Division Multiplexing* (OFDM)

OFDM merupakan bentuk khusus untuk FDM (*Frequency Division Multiplexing*). Pada FDM, suatu *bandwidth* tertentu dibagi menjadi beberapa kanal yang tersendiri. Agar tidak saling menginterferensi satu sama lain maka diberi jarak antar kanal (*guardband*), hal ini menyebabkan *bandwidth* yang digunakan tidak efisien. Sedangkan untuk OFDM, keempat kanal yang ada dalam satu *bandwidth* seakan-akan ditumpang tindihkan menjadi satu. Tidak ada jarak yang diberikan antar kanal (*guardband*). Dengan demikian, OFDM sangat efisien dalam penggunaan *bandwidth.*



Gambar 2.8 Spektrum *Frequency Division Multiplexing*



Gambar 2.9 Spektrum *Orthogonal Frequency Division Multiplexing* [5]

Spektrum frekuensi kanal pada OFDM dapat ditumpangtindihkan dan tidak terjadi saling interferensi antar kanal, hal ini terjadi karena masing-masing sinyal transmisi dalam setiap kanal bersifat saling *orthogonal* satu dengan yang lain. Kanal-kanal tersebut tidak akan saling menginterferensi karena bernilai nol atau saling meniadakan, hal ini disebabkan oleh setiap kanal yang berdekatan jatuh tepat pada titik tengah spektrum yang membawa informasi (spektrum yang memiliki *power* tertinggi). Untuk mengatur supaya kanal spektrum satu dengan yang lainnya jatuh tepat pada titik tengah spektrum yang membawa informasi, setiap sinyal transmisi pada setiap kanal harus bersifat saling *orthogonal*.[7]

2.4 Sinyal Ortogonal

Sebuah sinyal dapat diinterpretasikan sebagai sebuah vektor. Seperti halnya sebuah vektor, sinyal dapat dioperasikan dengan operasi-operasi vektor seperti penambahan dan perkalian dengan skalar ataupun dengan vektor lainnya.

Dengan demikian sinyal memenuhi sifat matematis dari ruang vektor. Jika ada dua buah vektor dan diintegralkan memenuhi persamaan dibawah ini :

$$\int_0^T S_i(t)S_j(t)dt \begin{cases} C & i = j \\ 0 & i \neq j \end{cases} \tag{2.1}$$

maka hasil perkalian antara dua fungsi tersebut akan bernilai nol jika basis vektor antara kedua fungsi tersebut berbeda dan perkalian antara kedua fungsi tersebut dikatakan *orthogonal*. Jika diterapkan pada transformasi Fourier, apabila ada suatu sinyal $x(t)$ dalam selang waktu $0 \leq t \leq T$ dan $x(t) = 0$ di tempat lain:

$$x(t) = \begin{cases} C & 0 \leq t \leq T \\ 0 & selainnya \end{cases} \tag{2.2}$$

maka dengan menggunakan transformasi Fourier dapat dituliskan sebagai:

$$x(t) = \frac{a_o}{2} + \sum_{k=1}^{\infty} a_k \cos(2\pi f_k t) - \sum_{k=1}^{\infty} b_k \sin(2\pi f_k t) \tag{2.3}$$

dengan a_k dan b_k adalah koefisien Fourier dengan:

$$a_k = \frac{2}{T}\int_0^T \cos(2\pi f_k t)x(t)dt \tag{2.4}$$

dan

$$b_k = -\frac{2}{T}\int_0^T \sin(2\pi f_k t)x(t)dt \tag{2.5}$$

di mana T adalah periode dari sinyal, t adalah waktu dan f_k adalah frekuensi dari gelombang sinusoidal dimana f_k = k / T adalah bilangan bulat (k = 0,1,2,3,4...). Jika dianalogikan ke dalam persamaan gelombang sinusoidal maka:

$$x(t) = A\sin\omega t \tag{2.6}$$

di mana A adalah amplitudo sedangkan sin ωt adalah vektor basis dari $x(t)$. Maka pada tranformasi Fourier koefisien Fourier (a_k dan b_k) adalah amplitudo dari gelombang kosinus dan (negatif) sinus dari frekuensi f_k, sedangkan gelombang kosinus dan (negatif) sinus adalah basis dari transformasi Fourier dalam selang waktu T. Setiap sinyal dapat dituliskan sebagai kombinasi *linear* dari basis-basisnya.

Bentuk transformasi Fourier di atas dapat dituliskan kembali ke dalam bentuk umum bahwa sebuah fungsi dibangun dari kombinasi *linear* dari basis-

basisnya dengan memandang sinyal $x(t)$ sebagai vektor berdimensi N ($\mathbf{x} \in \Re^N$) dalam vektor-vektor basis $\{\mathbf{v}_i\}_{i=1}^{N}$ sebagai berikut:

$$x = \sum_{i=1}^{N} \alpha_i \cdot v_i \tag{2.7}$$

koefisien α_i diperoleh dengan:

$$\alpha_i = v_i \cdot x \tag{2.8}$$

Basis-basis dari vektor yang digunakan disebut *orthonormal* jika antara dua basis vektor yang berbeda saling *orthogonal* satu sama lain. Dengan demikian, jika masing-masing basis vektor kita normalisasikan, maka perkalian titik dari dua basis vektor yang berbeda adalah nol. Sedangkan perkalian titik dari dua basis vektor yang sama memberikan nilai 1. Atau dapat dituliskan:

$$\boldsymbol{v}_i \cdot \boldsymbol{v}_k = \delta_{ik} \tag{2.9}$$

dimana δ_{ik} adalah *Kronecker Delta* ($\delta_{ik} = 0$ untuk $i \neq k$ dan $\delta_{ik} = 1$ untuk $i = k$).

Transformasi Fourier menguraikan sinyal ke dalam fungsi basisnya yaitu dalam bentuk kosinus dan (negatif) sinus dengan frekuensi $f_k = k/T$ di mana k adalah bilangan bulat sehingga untuk gelombang kosinus dan (negatif) sinus, gelombang-gelombang tersebut akan saling *orthogonal* karena frekuensinya saling berkelipatan.

Ini berhubungan dengan sifat bahwa dua sinyal sinusoidal saling *orthogonal* jika salah satu sinyal memiliki frekuensi sebesar kelipatan bulat dari frekuensi sinyal yang lainnya. Sedangkan untuk frekuensi yang sama, sinyal kosinus dan (negatif) sinus saling *orthogonal*. Karenanya kita dapat melihat transformasi Fourier sebagai penguraian suatu sinyal kepada basis-basis fungsi yang saling *orthonormal*.

Pada aplikasi nyata, sinyal yang digunakan adalah sinyal kompleks. Jika kita mempunyai sebuah sinyal kompleks $s(t)$ pada selang waktu $[0,T]$ dan nol pada daerah lain, maka kita dapat menuliskan transformasi Fourier untuk sinyal kompleks sebagai:

$$\mathrm{s(t)} = \sum_{k=-\infty}^{\infty} \alpha_k v_k(t) \tag{2.10}$$

dengan fungsi basis:

$$v_k(t) = \sqrt{\frac{1}{T}} \exp\left(j2\pi \frac{k}{T} t \right) \prod\left(\frac{t}{T} - \frac{1}{2} \right) \tag{2.11}$$

fungsi-fungsi basisnya saling orthonormal, atau dapat dituliskan:

$$\int_{-\infty}^{\infty} v_i^*(t) v_k(t) dt = \delta_{ik} \tag{2.12}$$

koefisien α_k didapat dengan:

$$\alpha_k = \int_{-\infty}^{\infty} v_i^*(t) s(t) dt \tag{2.13}$$

Untuk sinyal kompleks, fungsi basis yang dihasilkan dari transformasi Fourier adalah juga *orthonormal*. Dengan demikian, kita dapat melihat bahwa transformasi Fourier memiliki peranan yang besar dalam pembangkitan sinyal OFDM. [8]

2.5 Prinsip Kerja OFDM

Deretan data informasi yang akan dikirim dikonversikan ke bentuk paralel, sehingga bila *bit rate* semula adalah R, maka *bit rate* di tiap-tiap jalur paralel adalah R/M. M adalah jumlah jalur paralel (sama dengan jumlah *subcarrier*). Setelah itu, modulasi dilakukan pada tiap-tiap *subcarrier*. Modulasi bisa berupa BPSK, QPSK, QAM atau yang lain. BPSK, QPSK, dan QAM adalah beberapa teknik modulasi yang sering digunakan pada OFDM. Pada Skripsi ini digunakan modulasi QAM.

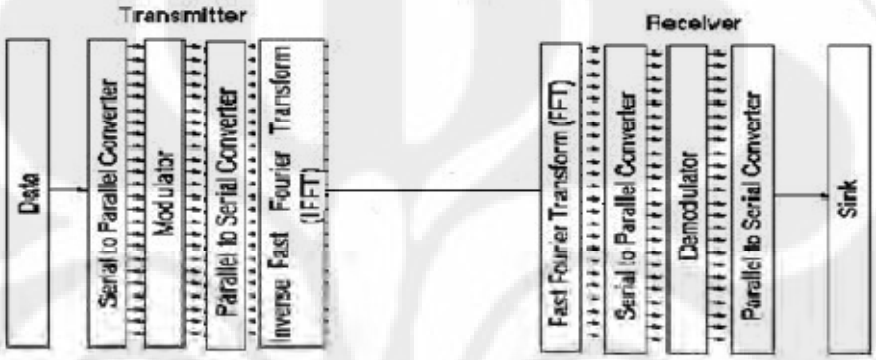
Sinyal yang telah termodulasi tersebut kemudian diaplikasikan ke *Inverse Discrete Fourier Transform* (IDFT) untuk pembuatan simbol OFDM dan mengubah *frequency domain* menjadi *time domain*. Setelah itu, simbol-simbol OFDM dikonversikan lagi kedalam bentuk serial kemudian sinyal dikirim. Sinyal yang dikirim dapat dinyatakan melalui persamaan 2.14.

$$s(t) = \mathrm{Re}\left\{ \sum_{n=-\infty}^{+\infty} b_n f(t - nT) e^{j(\omega_0 t + \varphi)} \right\} \tag{2.14}$$

Penggunaan IDFT ini memungkinkan pengalokasian frekuensi yang saling tegak lurus (*orthogonal*) dan mengubah domain sinyal dari *frequency domain* ke *time domain* untuk selanjutnya sinyal dikirim ke *receiver* OFDM. Penerapan IDFT dapat dinyatakan melalui persamaan 2.15.

$$X(nT) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\exp(jk2\pi\frac{n}{N}) \tag{2.15}$$

Sinyal OFDM dibangkitkan dan diproses secara dijital. Hal ini dimaksudkan untuk mengurangi kerumitan menyediakan sejumlah besar osilator. Oleh karena itu, proses modulasi dan demodulasi dilakukan dengan teknik pengolahan dijital, yakni *Discrete Fourier Transform* (DFT). Blok dasar OFDM yang lengkap dapat dilihat pada Gambar 2.10.



Gambar 2.10 Blok dasar OFDM [7]

2.6 Modulasi QAM [3]

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

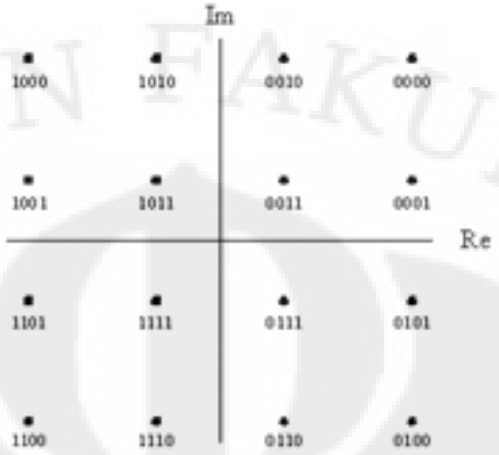
$$s(t) = I(t)\cdot\cos\omega_c t + Q(t)\cdot -\sin\omega_c t \tag{2.16}$$

Dengan:

$$\begin{aligned} I(t) &= A\cdot\cos\theta \\ Q(t) &= A\cdot\sin\theta \end{aligned} \tag{2.17}$$

Dari persamaan 2.16 dan 2.17, 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 *(ampliude 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°. *Amplitude* dan *phase* untuk masing-masing simbol pada QAM dapat digambarkan dalam sebuah

diagram dua dimensi yang disebut diagram konstelasi, seperti misalnya diagram konstelasi untuk 16-QAM yang, dapat dilihat pada Gambar 2.11.



Gambar 2.11 Diagram konstelasi 16 QAM

Sumbu horizontal merupakan sumbu yang mewakili cos ω_c dari persamaan dan disebut I *(imphase),* sedangkan sumbu vertikal adalah sumbu yang mewakili - sin ω_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 diteruskan pada proses selanjutnya.

2.7 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

ortogonalitas *subcarrier,* pada sisi pemancar dan penerima. Formula yang digunakan untuk FFT dan IFFT adalah:

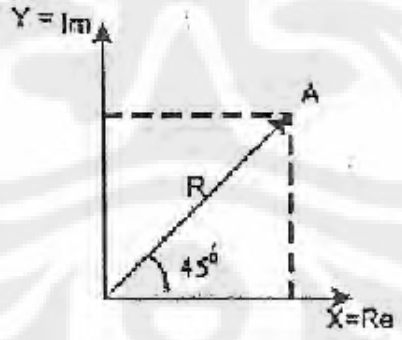
IFFT $$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k) e^{j2\pi kn/N}$$ k=0, 1, 2,…N-1 (2.18)

FFT $$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}$$ n =0, 1, 2,…N-1 (2.19)

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

$$e^{j2\pi f_n t} = \cos(2\pi f_n t) + j\sin(2\pi f_n t) \quad (2.20)$$

Secara analitis, harga dari proses transformasi Fourier dapat dilihat sebagai suatu bentuk kompleks yang mempunyai nilai riil (x) dan imajiner (y). Telah diketahui bahwa bilangan kompleks dapat dinyatakan dalam bentuk *polar,* yang terdiri dari *magnitude (R)* dan *phase (θ).* Hubungan yang menggambarkan antara bentuk *polar* dan *rectangular* dapat dilihat pada Gambar 2.12.

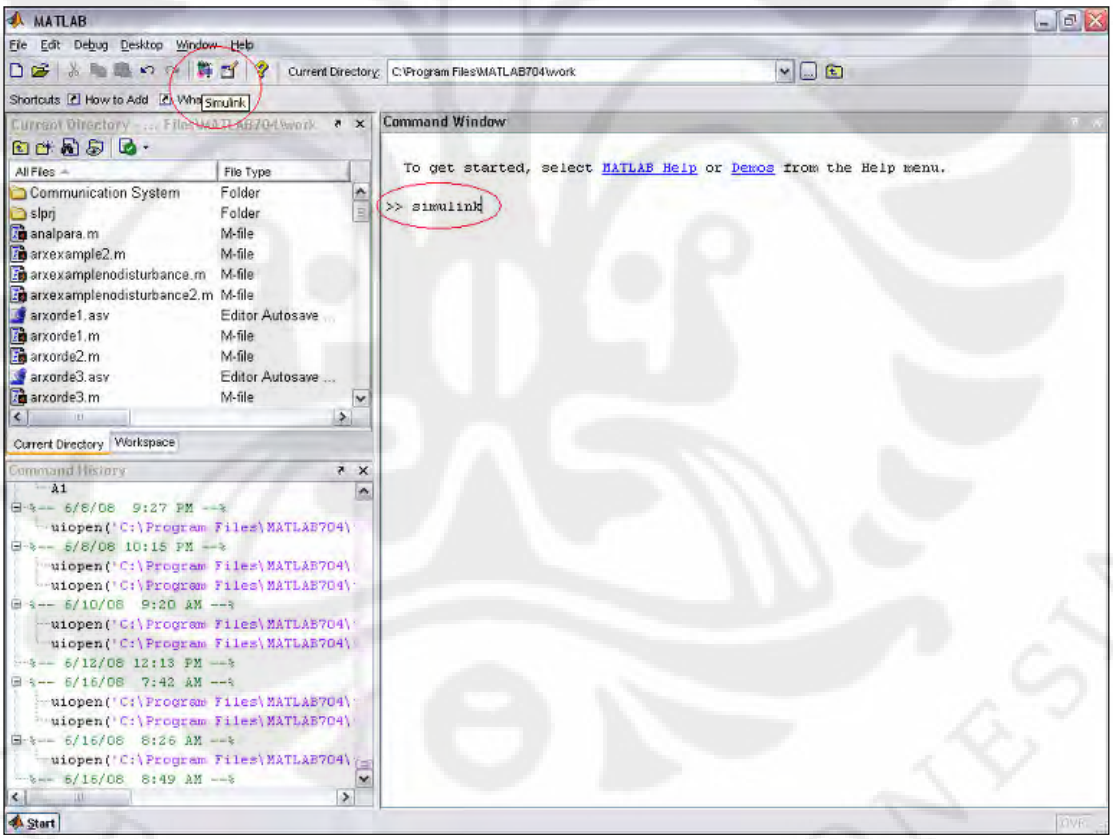


Gambar 2.12 Hubungan bentuk *polar* dan *rectangular* [3]

BAB 3
RANCANG BANGUN

3.1 Simulink

Simulink adalah perangkat lunak sub-program dari MATLAB yang biasa digunakan untuk pemodelan, simulasi dan analisa sistem. Dengan menggunakan perangkat lunak ini kita dapat membangun blok-blok model untuk mensimulasi sistem yang kita kehendaki. Cara menggunakan Simulink hanya dengan mengetikkan perintah "*simulink*" pada *command window* dari MATLAB. Cara lain adalah cukup dengan menekan tombol simulink pada *toolbar* MATLAB seperti pada Gambar 3.1.



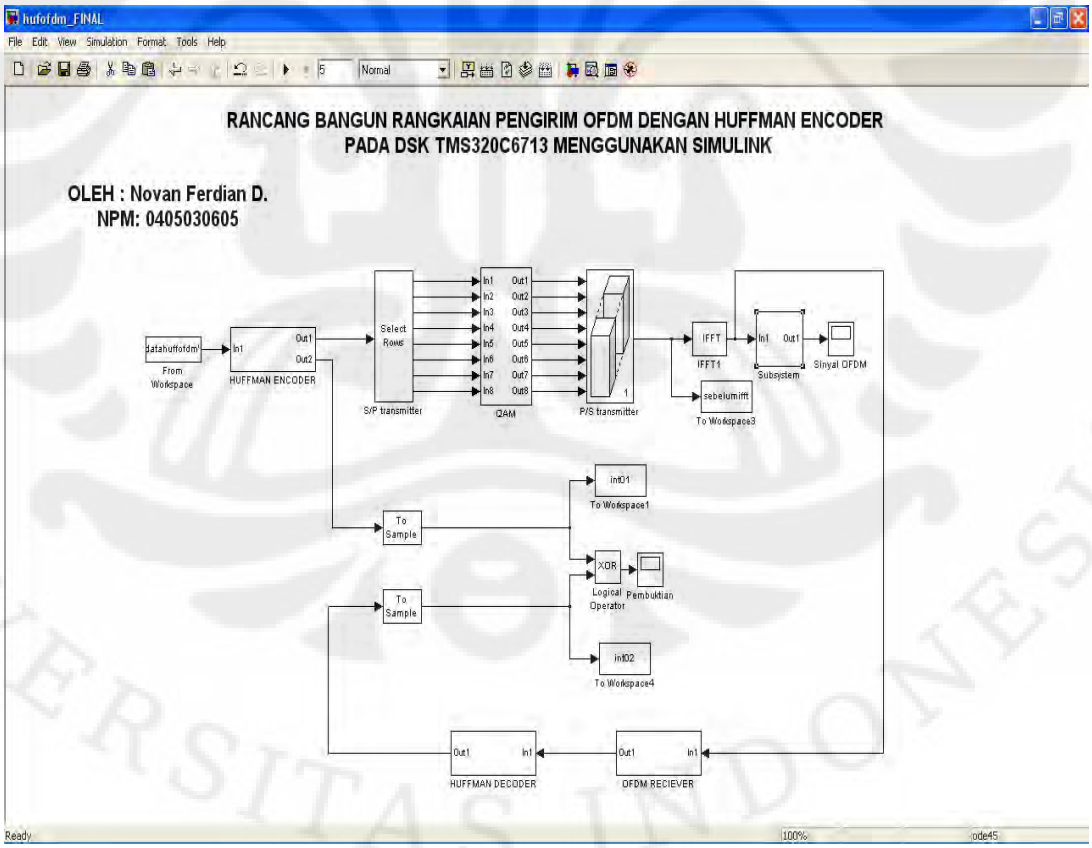
Gambar 3.1 Cara-cara menjalankan Simulink

Dengan menjalankan program ini, kita akan diberikan pilihan blok-blok model yang dapat kita gabungkan satu dengan yang lain untuk membentuk simulasi sistem yang kita kehendaki.

Dari Simulink, kita juga dapat menghubungkannya dengan perangkat luar seperti DSK TMS320C6713 dengan cara yang mudah. Simulink akan membangkitkan secara otomatis kode bahasa pemrograman yang digunakan pada DSP tersebut.

3.2 Blok Diagram Rangkaian Pengirim OFDM Dengan *Huffman Encoder* Pada Simulink

Rancang bangun model rangkaian pengirim OFDM dengan *Huffman code* dibuat dengan menggunakan Simulink yang terdapat pada MATLAB. Model rangkaian yang telah dibuat dapat dilihat pada Gambar 3.2.



Gambar 3.2 Model rangkaian pengirim OFDM dengan *Huffman code* pada Simulink

Berikut adalah penjelasan blok-blok yang digunakan:

1. *From Workspace*

   Blok *from workspace* adalah blok yang digunakan sebagai sumber data (*source*) yang merupakan sumber informasi yang akan dimodulasikan pada pemancar OFDM di mana sebelum masuk ke pemancar OFDM keluaran dari blok *from workspace* ini akan dikodekan terlebih dahulu oleh blok *Huffman encoder* dengan menggunakan *Huffman coding*. Keluaran dari blok ini adalah berupa matriks ukuran 1000 x 1 yang berisikan bilangan bulat acak dari angka 0 sampai 63 yang telah dipilih sebagai data.

2. *Huffman Encoder*

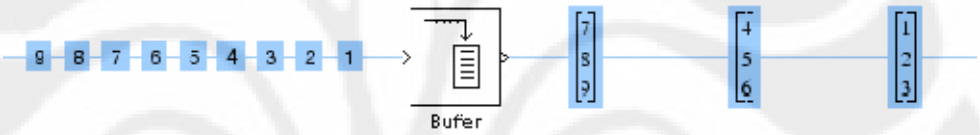
   Di dalam blok ini terdapat blok *Embedded MATLAB Function* yang dapat berisikan perintah-perintah Matlab dengan inputan dan keluarannya sesuai dengan keinginan user. Aliran data input yang berasal dari data *source* akan masuk ke *Huffman Encoder.* Pada proses *Huffman Encoder* pertama akan di cari nilai probabilitas masing-masing simbol yang masuk lalu kemudian dengan fungsi "huffmandict" akan dibentuk 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 yang sama dengan 1. Keluaran dari fungsi "huffmandict" ini adalah berupa kamus (*codeword*) untuk masing-masing simbol. Lalu kamus ini akan menjadi input bagi fungsi "huffmanenco" yang akan menghasilkan data output berupa data dari *source* yang telah dikodekan dalam bentuk biner dengan *Huffman coding*.

3. Bit *to Integer Converter*

Bit *to Integer Converter* berfungsi mengubah nilai *binary* setiap beberapa bit tertentu sesuai dengan keinginan menjadi satu nilai *integer*. Pada Skripsi ini, setiap enam bit diubah menjadi satu nilai *integer*.

4. *Buffer*

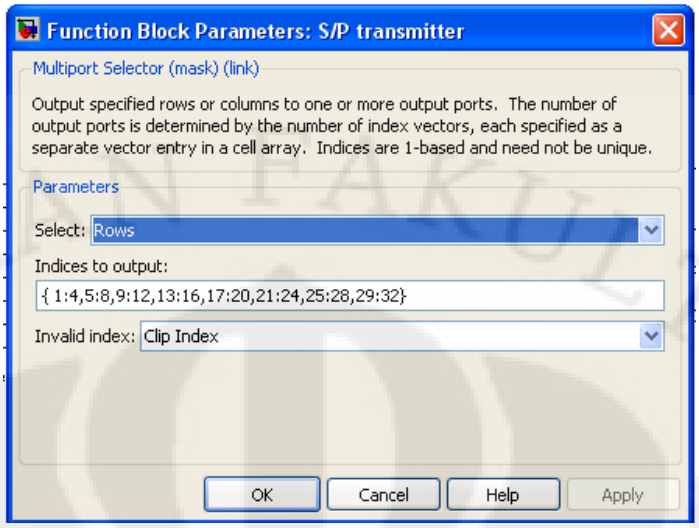
Buffer dapat digunakan untuk mengubah *sample based* menjadi *frame based* dengan *data rate* yang bisa lebih tinggi atau rendah sesuai dengan keinginan. Perubahan *sample based* menjadi *frame based* yang dilakukan *buffer* dapat dilihat pada Gambar 3.3.



Gambar 3.3 *Buffer*

5. *Multiport Selector*

Dalam blok *Multiport Selector* mode yang digunakan adalah *Select rows*. *Select rows* digunakan untuk mengubah data serial menjadi paralel. *Select rows* diisi dengan nomor urut data yang ingin diambil sesuai dengan keinginan untuk digunakan pada proses selanjutnya. Data serial berupa nilai akan diubah menjadi data paralel dengan menggunakan blok ini. Data serial berupa nilai *integer* dalam satu *frame* akan diubah menjadi delapan data paralel dengan menggunakan blok *select rows* sehingga masing-masing *bus* menerima empat buah nilai *integer*. Konfigurasi parameter *multiport selector* ditunjukkan oleh Gambar 3.4.



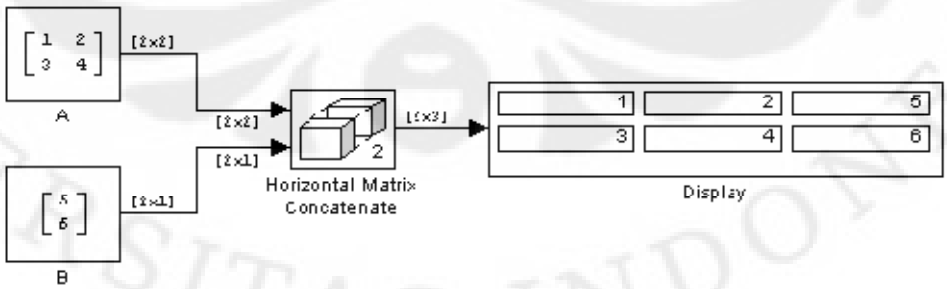
Gambar 3.4 Konfigurasi parameter *Multiport Selector*

6. QAM

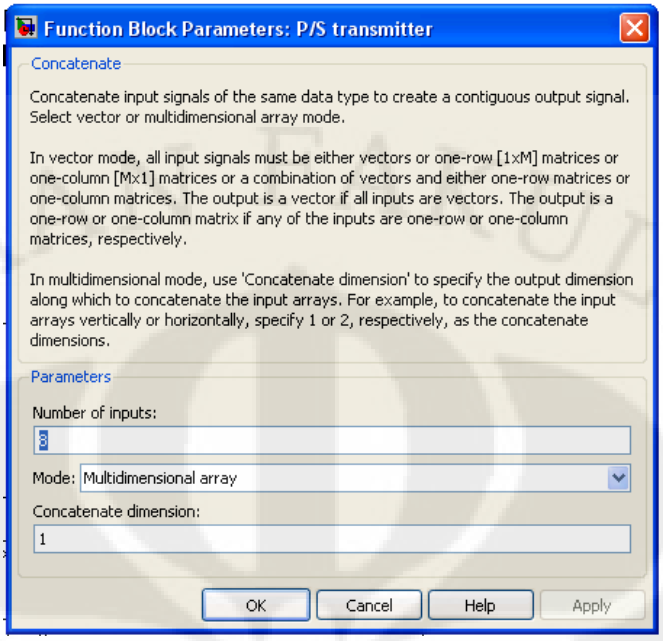
Blok QAM digunakan untuk memodulasi sinyal secara QAM yang terdapat di sisi *transmitter*.

7. *Matrix Concatenation*

Matrix Concatenation digunakan untuk mengubah data paralel menjadi data serial. *Matrix Concatenation* yang digunakan dibuat menjadi *mode multidimensional array* dengan 8 *input* dan 1 *output*. Prinsip kerja *Matrix Concatenation* dapat dilihat pada Gambar 3.5. Konfigurasi parameter blok *concatenate* ditunjukkan oleh Gambar 3.6



Gambar 3.5 *Horizontal matrix concatenate*



Gambar 3.6 Konfigurasi parameter *matrix concatenate*.

8. *To Workspace*

   To Workspace berfungsi menyimpan hasil simulasi ke dalam *workspace*

9. IFFT

   Blok IFFT berfungsi melakukan transformasi dari *domain* frekuensi ke waktu.

10. *Outport*

    Outport digunakan untuk mengirim data ke luar dari suatu *subsystem.*

11. *To Frame*

    To Frame digunakan untuk mengubah *sample based* menjadi f*rame based.*

12. *To Sample*

    To Sample digunakan untuk mengubah *frame based* menjadi *sample based.*

13. *Spectrum Scope*

Spectrum scope digunakan untuk melihat spektrum dari suatu sinyal.

14. XOR

Hasil keluaran dari blok *input* data yang telah diubah menjadi bit dengan menggunakan *integer to* bit *converter* di-XOR-kan, artinya apabila bit yang bertemu antara bit yang ada di *input* data dengan bit berlainan (0 dan 1) maka keluaran dari nlok XOR tersebut adalah bernilai 1. Berikut tabel 3.1 diagram XOR :

Tabel 3.1 Diagram XOR

A	B	C
Zero	Zero	0
Zero	Nonzero	1
Nonzero	Zero	1
Nonzero	Nonzero	0

17. *Complex to Real Imaginary*

Blok Complex to real imaginary digunakan untuk memecah nilai *complex* menjadi *real* dan *imaginary.*

18. *Sine Wave*

Sine wave digunakan untuk menghasilkan gelombang *sinusoidal* dengan frekuensi dan ampitudo tertentu

19. *Rate Transition*

Rate transition berfungsi meminimalisasi *delay* data antara dua atau lebih *port* dengan *data rate* berbeda.

20. *Product*

Product digunakan untuk mengalikan dua buah sinyal.

21. *Add*

Add digunakan untuk menjumlahkan dua buah sinyal *input.*

22. *Time Scope*

Time scope digunakan untuk menampilkan keluaran dalam *time domain* yang berbentuk *sample based.*

24. *Subsystem*

Subsystem digunakan untuk mengelompokkan suatu *model* menjadi suatu *subsystem.*

25. DAC DSK 6713

DAC DSK 6713 dipakai agar dapat menggunakan DAC dari DSK 6713

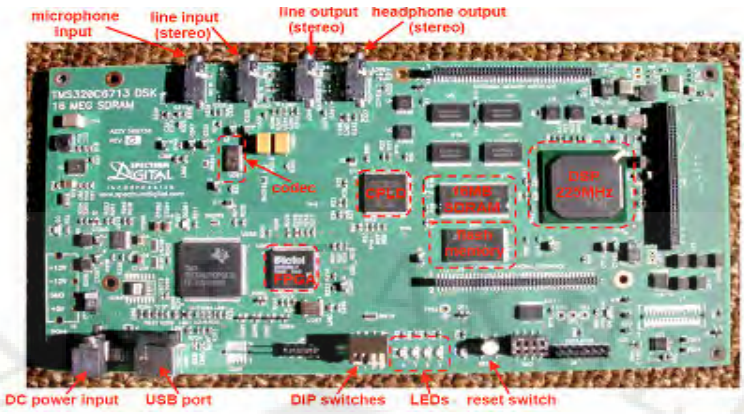
.

26. C6713 *Board*

C6713 *board* dipakai agar kita dapat melakukan *targetting* ke DSK *board* dengan Simulink

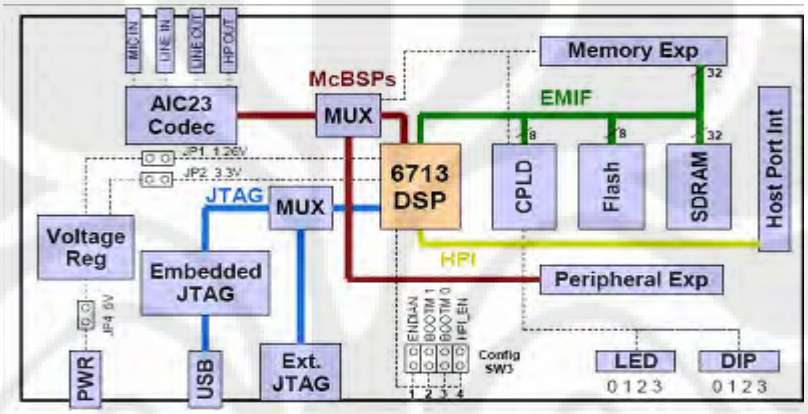
3.3 *Digital Signal Processor Starter Kit* TMS320C6713

Blok diagram sebagaimana dibuat dalam Simulink telah siap untuk diimplementasikan pada sebuah papan *processor*. Pada rancang bangun ini, papan *processor* yang digunakan adalah DSP Starter Kit TMS320C6713. *Digital Signal Processor* yang dalam hal ini adalah TMS320C6713 adalah *mikroprosesor* yang dibuat dalam arsitektur khusus untuk digunakan sebagai pemroses sinyal dijital (*Digital Signal Processing*). Modul ini biasa digunakan untuk membuat sebuah rancang bangun dasar dari sebuah pemroses sinyal yang hendak disimulasikan. Papan TMS320C6713 ini dapat kita lihat pada Gambar 3.7 berikut:



Gambar 3.7 Papan DSK TMS320C6713

dengan blok diagram ditunjukkan pada Gambar 3.8 berikut:

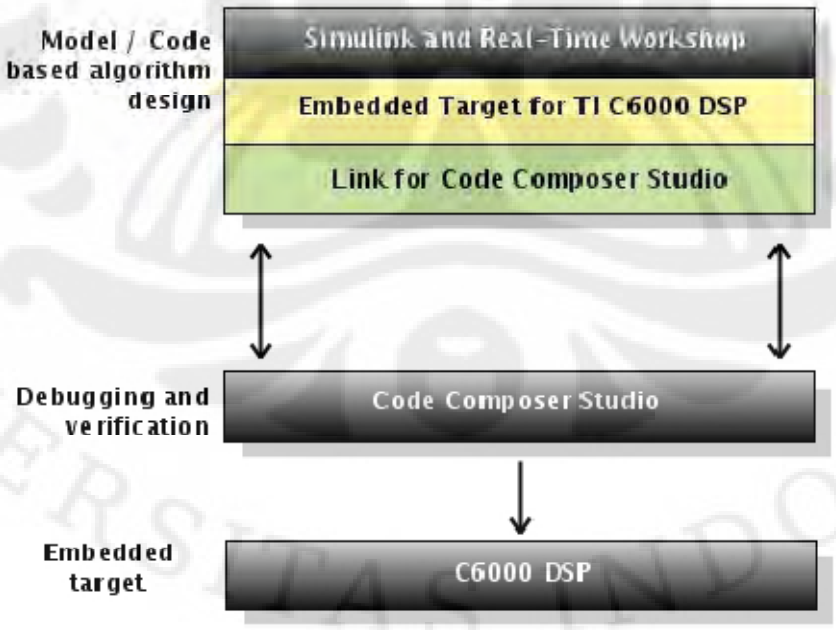


Gambar 3.8 Blok diagram dari DSK TMS320C6713

Beberapa bagian dari papan DSK ini yang perlu diperhatikan antara lain adalah LINE IN dan LINE OUT yang merupakan modul masukan dan keluaran dari DSK ini. Modul lain yang perlu diperhatikan adalah *Joint Team Action Group* (JTEG) yang memungkinkan terjadinya komunikasi antara DSK dengan komputer. Komunikasi yang dilakukan antara komputer dengan DSK dilakukan melalui port USB (*Universal Serial Bus*). Melalui port ini dilakukan tukar menukar data antara komputer dengan DSK dan juga sinyal-sinyal pengendalian dari komputer ke DSK. Proses komunikasi secara *real time* antara komputer dengan DSK ini disebut sebagai RTDX (Real Time Data Exchange). Modul lain yang juga dapat digunakan sebagai keluaran sinyal adalah modul LED dan DIP *switch*. Pada modul LED, sinyal keluaran direpresentasikan dengan nyala dari lampu LED. Sedangkan pada DIP switch, sinyal keluaran direpresentasikan dengan perubahan nilai tegangan dari tombol DIP *switch*.

3.4 Implementasi Blok Simulink Pada TMS320C6713

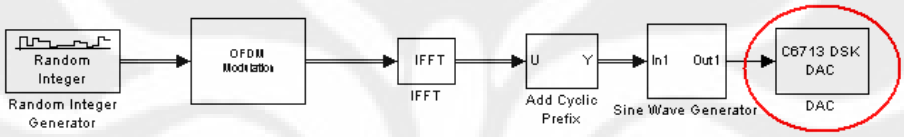
Blok yang telah dibuat pada Simulink dapat kemudian kita masukkan dalam DSK TMS320C6713 dengan hanya melakukan beberapa *setting* pada *Configuration Parameters*. Proses debugging dari Simulink ke DSK memerlukan perangkat lunak lain yaitu Code Composer Studio (CCS) yang merupakan sebuah komposer bahasa C/C++ yang digunakan untuk memrogram papan DSK dengan menggunakan bahasa C/C++. Maka sebenarnya yang dilakukan oleh Simulink adalah membangun kode pemrograman dalam bahasa C/C++ dari model yang disimulasikan. Kode pemrograman tersebut kemudian dijalankan oleh CCS. Dan kemudian Simulink dan papan DSK TMS320C6713 dapat berkomunikasi melalui kode-kode pemrograman yang dibuat oleh Simulink. Inilah kemudahan yang diberikan oleh Simulink, yaitu kita tidak perlu membuat dan membangun kode pemrograman dari model yang kita simulasikan. Kita hanya cukup membangun blok model pada Simulink dan dengan mudah kita akan mendapatkan kode pemrograman dalam bahasa C/C++ yang dapat digunakan untuk membuat rancang bangun dari model yang telah kita buat dalam Simulink pada DSK TMS320C6713. Gambar 3.9 menunjukkan algoritma dari penerapan Simulink pada DSK.



Gambar 3.9 Algoritma penanaman model ke dalam TMS320C6713 sebagai *embedded target*

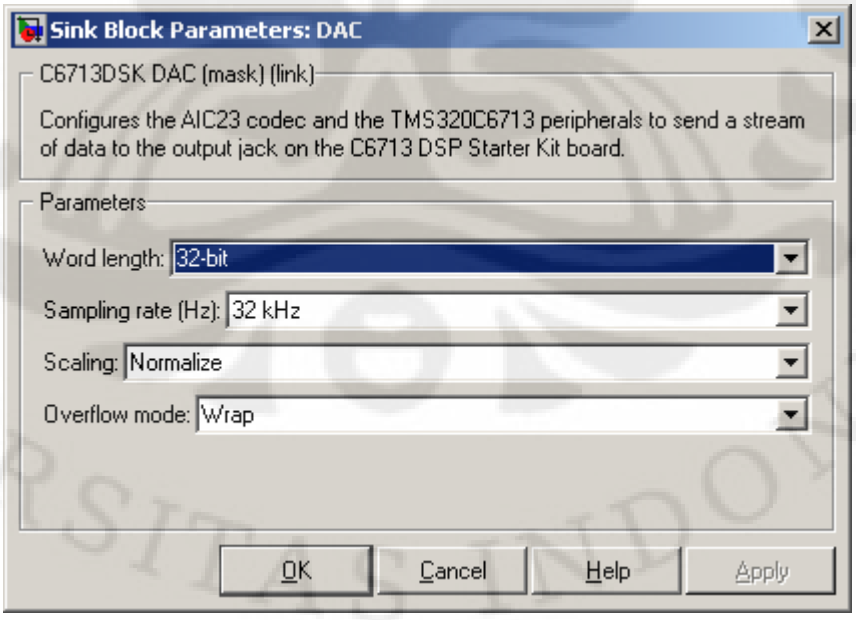
Berikut adalah langkah-langkah yang dilakukan untuk dapat mengimplementasikan model yang telah kita buat dalam Simulink ke dalam papan TMS320C6713 sebagai *embedded target*:

1) Memasangkan Digital-to-Analog Converter (DAC) sebagai keluaran dari DSK (Gambar 3.10). Pada blok ini, sinyal keluaran dari OFDM kemudian akan dikonversikan dari bentuk dijital ke dalam bentuk analog yang akan menjadi keluaran dari papan DSK. Dengan menggunakan blok ini, kita akan dapat melihat sinyal keluaran OFDM dari keluaran papan (LINE OUT) berupa sinyal kontinu.



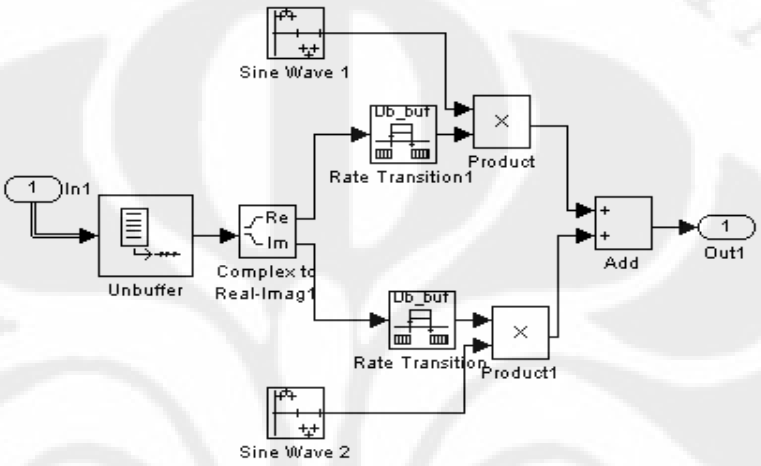
Gambar 3.10 Penambahan blok C6713 DSK DAC

Dengan parameter blok DAC seperti ditunjukkan pada Gambar 3.11 berikut:



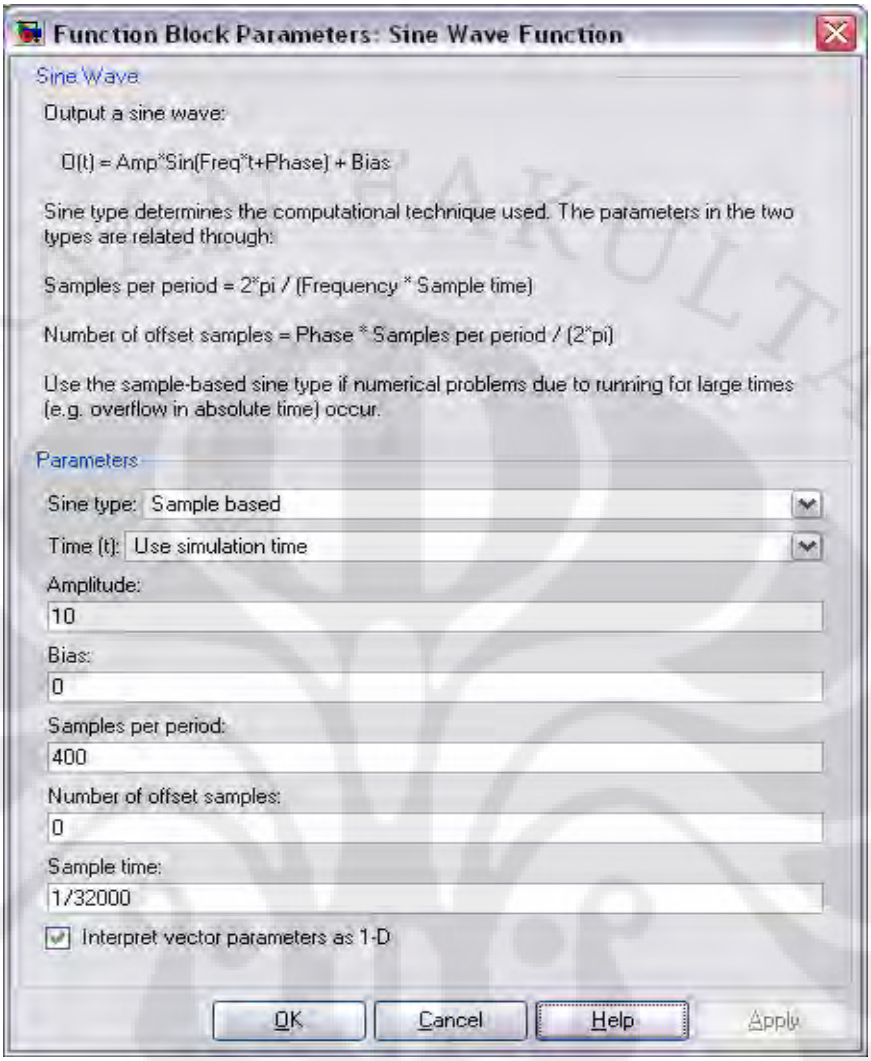
Gambar 3.11 Parameter blok C6713 DSK DAC

Pada Gambar 3.10 ada tambahan blok *sine wave generator*. Blok ini berfungsi untuk membangun sinyal sinusolidal agar keluaran dari modul simulasi dapat kita lihat dan ukur sebagai sinyal sinusoidal. Blok ini adalah sebuah *subsystem* yang terdiri dari beberapa blok dasar seperti yang dapat kita lihat pada Gambar 3.12 sebagai berikut:



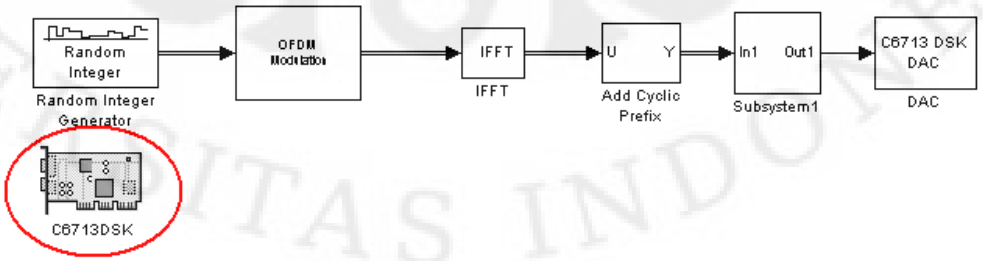
Gambar 3.12 Subsistem *sine wave generator*

Hal yang perlu diperhatikan dari subsistem ini adalah parameter blok dari *sine wave* 1 dan *sine wave* 2. Gambar 3.13 menunjukkan parameter dari blok *sine wave* 1 dan *sine wave* 2 sebagai berikut:



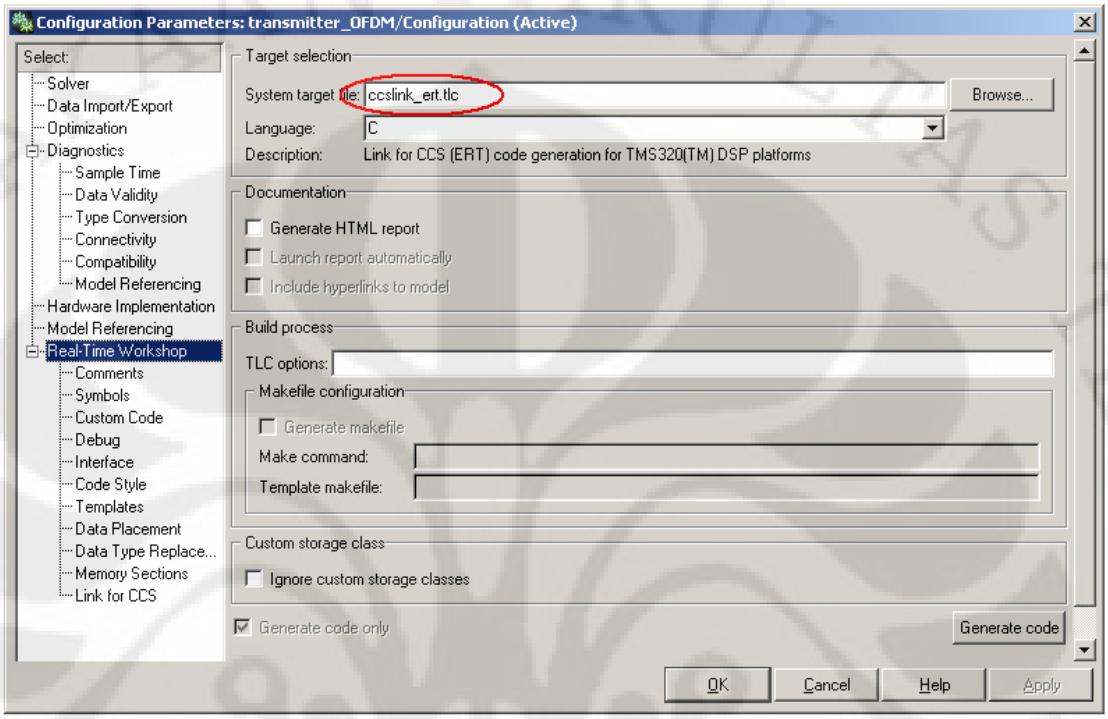
Gambar 3.13 Parameter blok *sine wave* 1 dan 2

2) Mencatumkan blok C6713DSK dari *library browser* ke model yang telah dibuat (Gambar 3.14). Blok ini menandakan bahwa simulasi yang kita buat siap untuk diimplementasikan pada papan DSK TMS320C6713 sebagai *embedded target*.

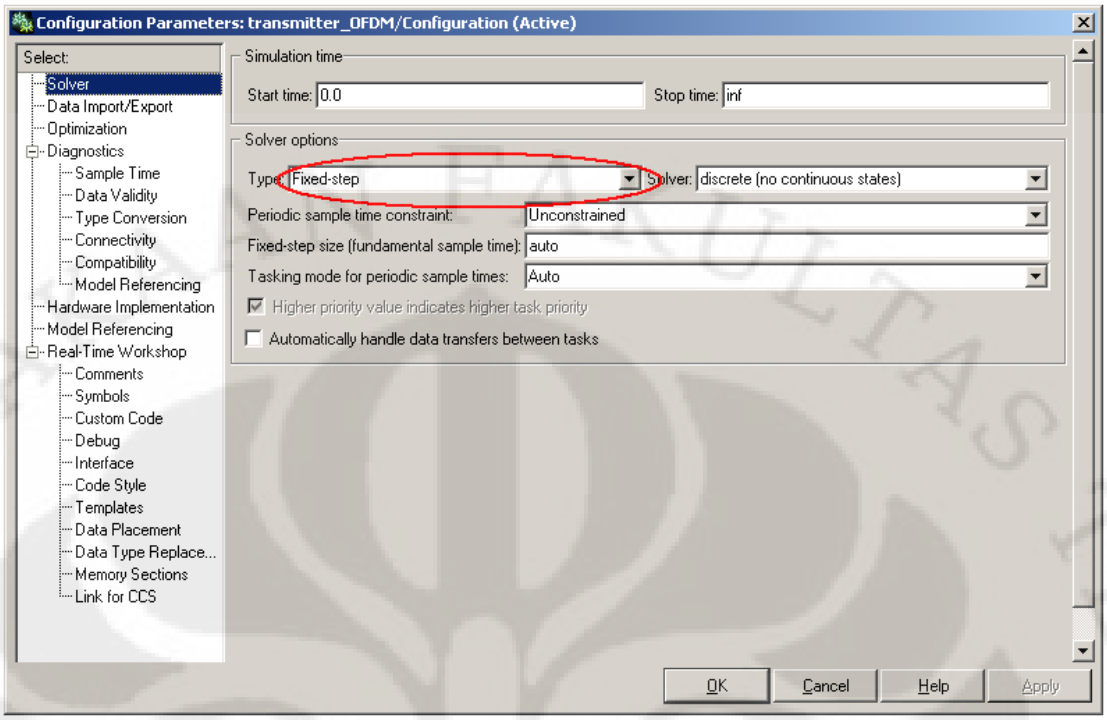


Gambar 3.14 Penambahan blok C6713 DSK

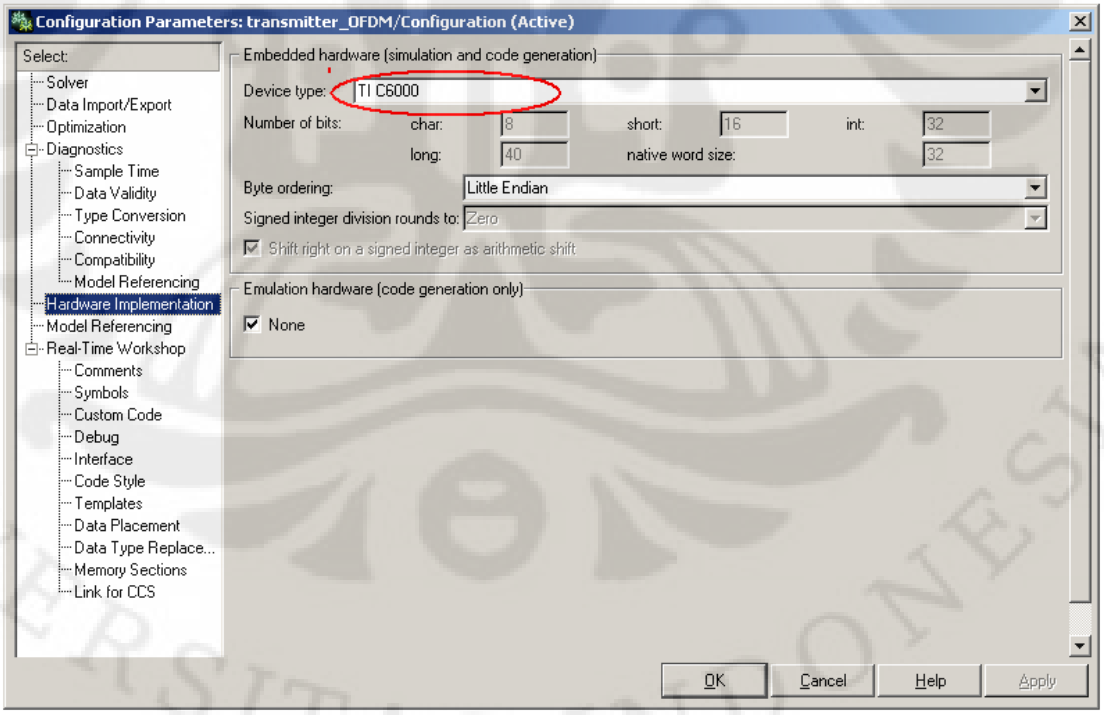
3) Pada *toolbar* pilih *simulations* → *configuration parameters* atau dengan menekan Ctrl+E. Dan lakukan beberapa *setting* seperti terlihat pada Gambar 3.15 sampai dengan Gambar 3.18 berikut:



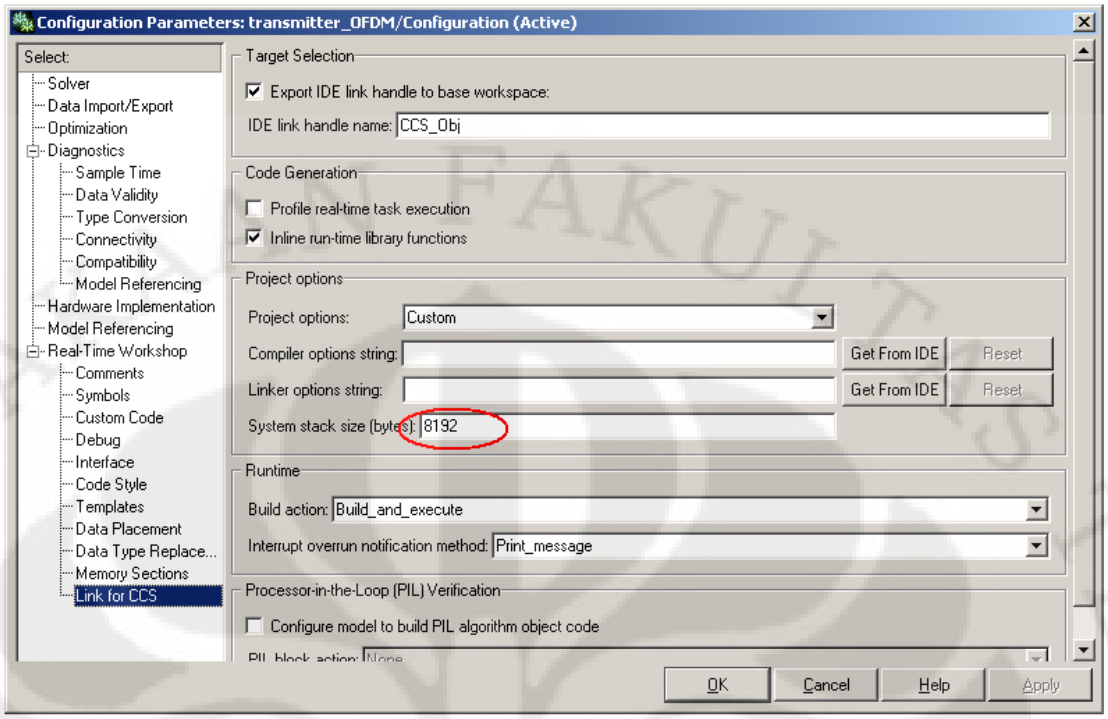
Gambar 3.15 Pada *tab* menu "*Real Time Workshop*"



Gambar 3.16 Pada *tab* menu "Solver"



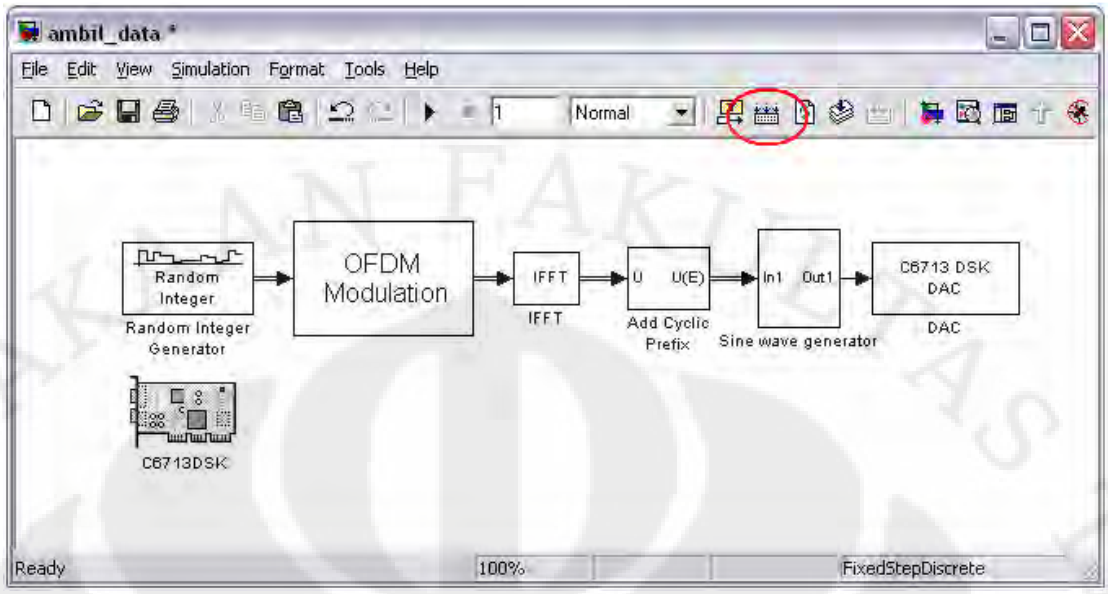
Gambar 3.17 Pada *tab* menu "Hardware Implementation"



Gambar 3.18 Pada *tab* menu "Link for CCS"

Setelah semua setting pada *Configuration Parameters* dilakukan kemudian pilih OK dan kembali ke model simulasi.

4) Pada *toolbar* Simulink, tekan tombol *incremental build* seperti pada Gambar 3.19. Setelah itu Simulink akan membuat kode bahasa C dari blok model yang telah dibuat dan dengan demikian blok model telah siap diimplementasikan pada papan DSK.



Gambar 3.19 Tombol "incremental build"

Setelah semua langkah tersebut dilakukan, maka Simulink akan membangun kode-kode pemrograman dalam bahasa C pada *Code Composer Studio*. Setelah proses pembuatan kode selesai, maka CCS akan membuka secara otomatis dan DSK telah terhubung dengan komputer melalui perangkat lunak CCS. Setelah proses ini selesai, maka pada DSK telah dimasukkan modul simulasi pemancar OFDM. Model ini kemudian akan berjalan pada perangkat DSK secara *real time*.

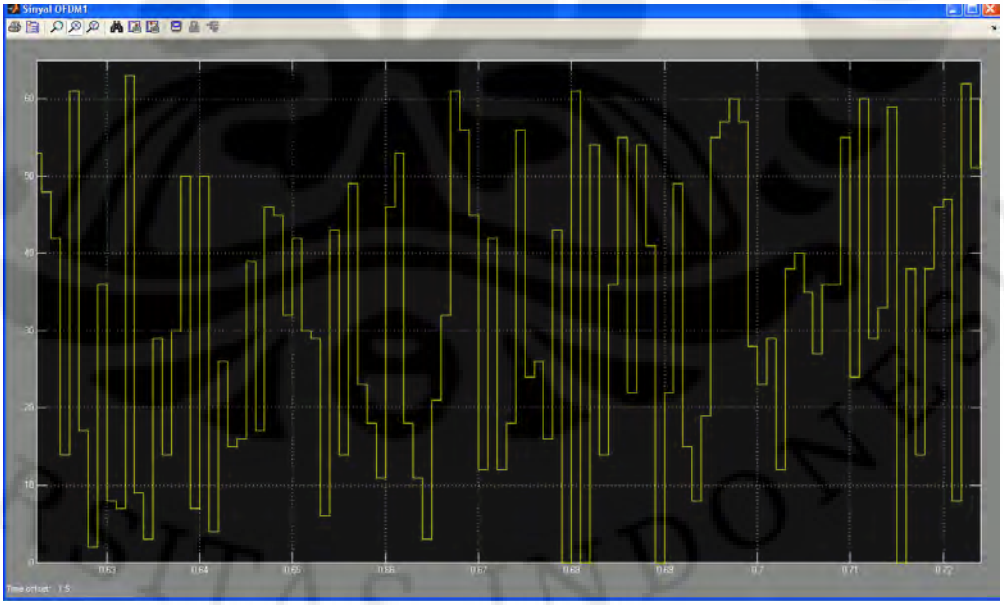
BAB 4
UJI COBA DAN ANALISA

4.1 Prosedur Uji Coba Dan Pengambilan Data

Pengujian terhadap model rangkaian pengirim OFDM dengan *Huffman code* yang telah dibuat terdiri dari dua metode, yakni pengujian secara simulasi dengan menggunakan simulink dan pengujian dengan menggunakan alat yaitu DSK TMS320C6713. Pengujian akhir dilakukan dengan cara membandingkan data yang telah diproses di *transmitter* dengan data yang dikirim dari *reciever*.

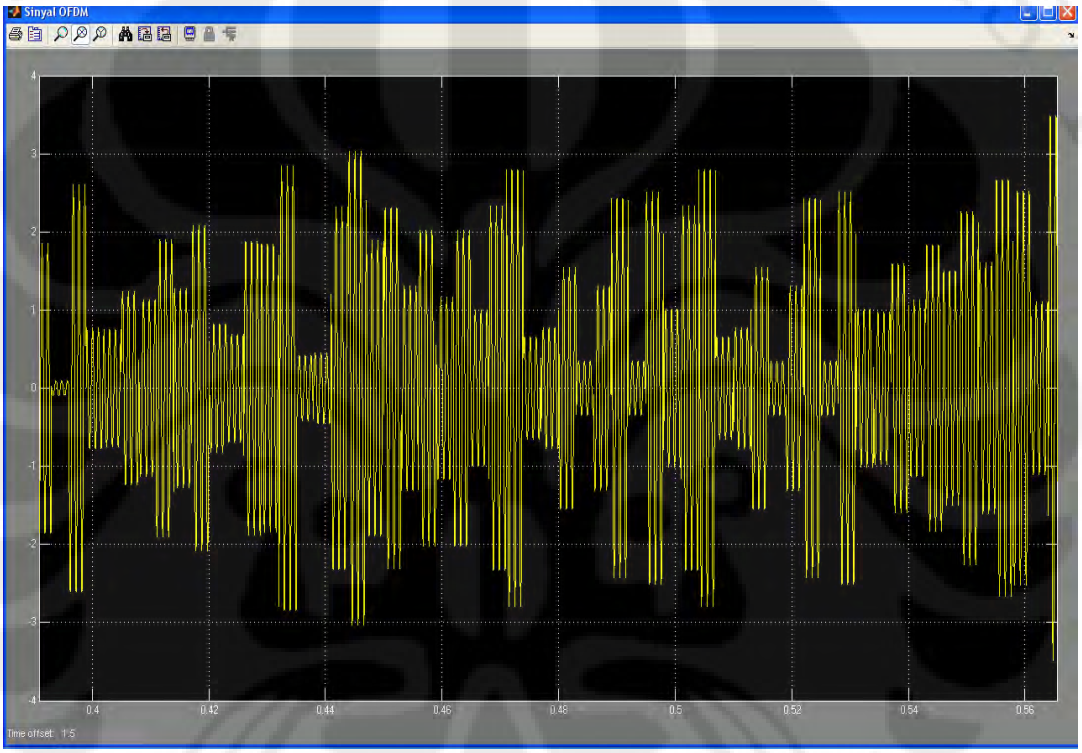
4.2 Simulink

Pada uji coba Simulink ini, data diambil dari sisi *transmitter* untuk dibandingkan dengan data yang di dapat dari sisi *receiver*. Pada sisi *transmitter*, rangkaian terdiri dari *Huffman encoder*, IFFT, *serial to paralel converter, modulator,* dan *paralel to serial converter*. Sinyal pada sisi *receiver* berasal dari rangkaian *transmitter* OFDM. Pada sisi *transmitter* data dibangkitkan dengan menggunakan blok *From Workspace* sehingga data yang dikirim hanya berupa angka bulat acak antara 0 sampai 63 seperti pada Gambar 4.1.



Gambar 4.1 Data kirim *From Workspace*

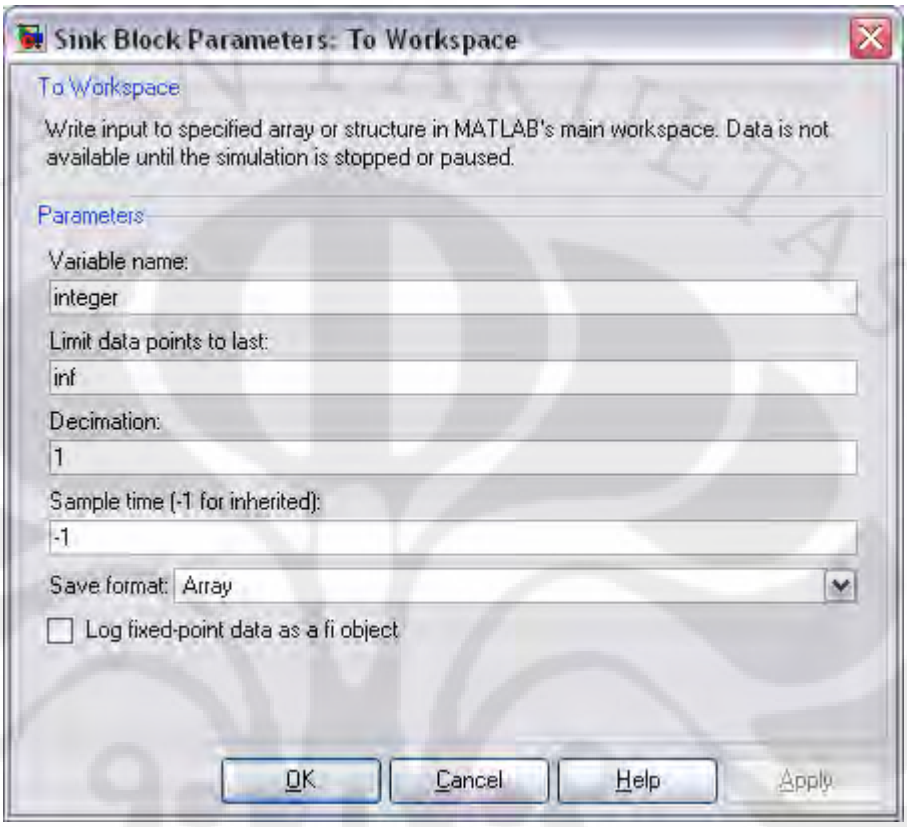
Selanjutnya data yang dibangkitkan oleh *blok From Workspace* menuju blok *Huffman encoder* untuk dilakukan proses kompresi data yang asli. Kemudian data yang telah dikodekan ini akan menuju blok modulasi QAM untuk dilakukan proses modulasi. Setelah dilakukan proses modulasi, kemudian sinyal menuju blok IFFT untuk proses pembentukan sinyal OFDM dan transformasi dari *domain* frekuensi ke *domain* waktu sehingga bentuk gelombang OFDM setelah melewati IFFT akan terlihat seperti Gambar 4.2.



Gambar 4.2 Sinyal OFDM

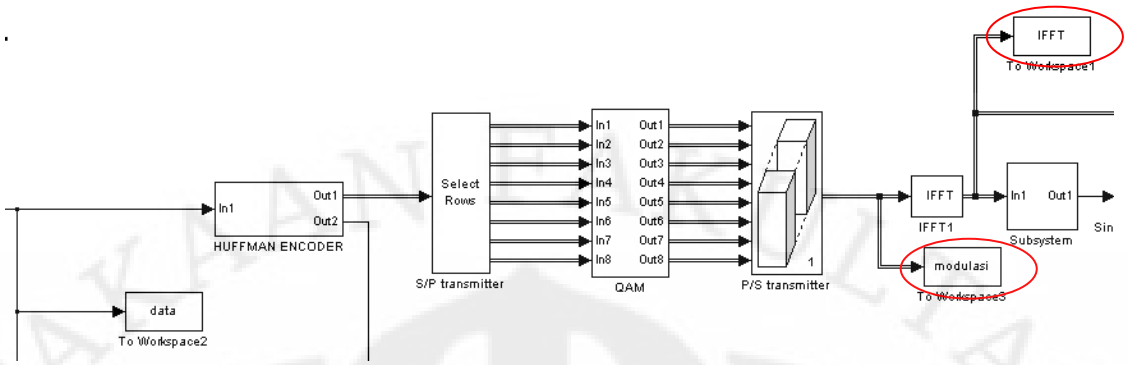
Pengukuran pada Simulink dapat ditelusuri sampai kepada bilangan-bilangan untuk masing-masing blok. Dengan mengambil nilai masukkan dan keluaran dari masing-masing blok, analisa dilakukan dengan melihat hubungan antara masukan dengan keluaran pada masing-masing blok. Pengambilan data untuk masing-masing blok dilakukan dengan menambahkan blok '*to workspace*' untuk mengambil data. Dengan menggunakan blok '*to workspace*' data yang diambil dari masing-masing blok diolah menjadi variabel pada Workspace dari MATLAB dan dapat langsung dilihat nilai masukan dari masing-masing variabel

pada MATLAB. Parameter blok '*to workspace*' ditunjukkan pada Gambar 4.3 berikut:

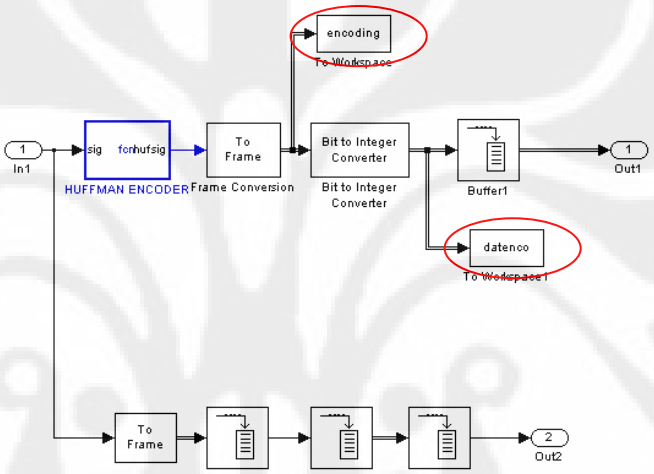


Gambar 4.3 Parameter blok 'to workspace'

Hal yang perlu diperhatikan dari parameter blok ini adalah pada baris '*Save format*', maka format yang dipilih adalah array. Data-data yang didapatkan dari blok to workspace kemudian disimpan dalam bentuk *array* pada MATLAB. Nama-nama variabel yang digunakan seperti yang dapat dilihat pada Gambar 4.4 dan Gambar 4.5 antara lain: data, encoding, dataenco, modulasi, dan IFFT.



Gambar 4.4 Pencantuman blok-blok '*to workspace*' pada modul utama



Gambar 4.5 Pencantuman blok '*to workspace*' pada *subsystem* Huffman encoder

4.2.1 Variabel 'data'

Data adalah nama variabel yang digunakan untuk menyimpan data keluaran dari *From workspace*. Nilai-nilai dari variabel ini bertipe data *integer* sebagaimana sumber data yang digunakan adalah sumber data integer. Rentang nilai dari variabel ini adalah antara 0 sampai dengan 63. Hal ini dikarenakan modulasi yang kita gunakan adalah 64-QAM. Setelah keluar dari *Huffman encoder*, variabel ini kemudian menjadi masukkan bagi blok 64-QAM yang akan memetakan bilangan sumber data *integer* menjadi bilangan kompleks. Dimensi dari variabel ini adalah *array* dengan dimensi [1000x1]. Telah dibahas pada bab sebelumnya bahwa untuk dapat memahami konsep ortogonalitas maka kita harus

memandang sinyal sebagai vektor. Pada penelitian ini lebar data yang digunakan untuk satu frame adalah sebesar 1000.

Jika kita menggunakan tipe data *integer* dengan rentang nilai antara 0 - 63, maka kita memerlukan 6 bit bilangan biner untuk masing-masing bilangan *integer* yang dibangkitkan. Pada parameter blok '*source*' pada bagian *Samples per frame* nilainya adalah 1000. Ini berarti pada selama satu detik dibangkitkan 1000 data *integer.*

4.2.2 Variabel 'Modulasi'

Variabel ini berisi nilai-nilai hasil *mapping* dari bilangan *integer* masukan 64-QAM. Tipe data untuk variabel ini adalah tipe data kompleks. Masing-masing bilangan *integer* masukan dari 64-QAM dipetakan menjadi bilangan kompleks yang bersesuaian. Pada variabel ini, ukuran vektor menjadi [32x1], jadi model ini memakai 32-point IFFT.

Dapat diketahui bahwa untuk FFT *length* yang lebih besar, spektrum frekuensi pada rentang frekuensi yang digunakan lebih baik, namun *side band* yang terjadi juga menjadi lebih besar.

4.2.3 Variabel 'IFFT'

Variabel modulasi seperti telah dijelaskan sebelumnya kemudian menjadi masukkan bagi blok IFFT. Pada blok ini dikerjakan perhitungan *Inverse Fourier Transform*, sehingga variabel 'IFFT' yang adalah keluaran dari blok 'IFFT' adalah nilai *Inverse Fast Fourier Transform* dari variabel 'modulasi'. Perhitungan IFFT melalui MATLAB untuk variabel 'modulasi' memberikan hasil yang sama dengan variabel 'IFFT' pada *workspace'*.

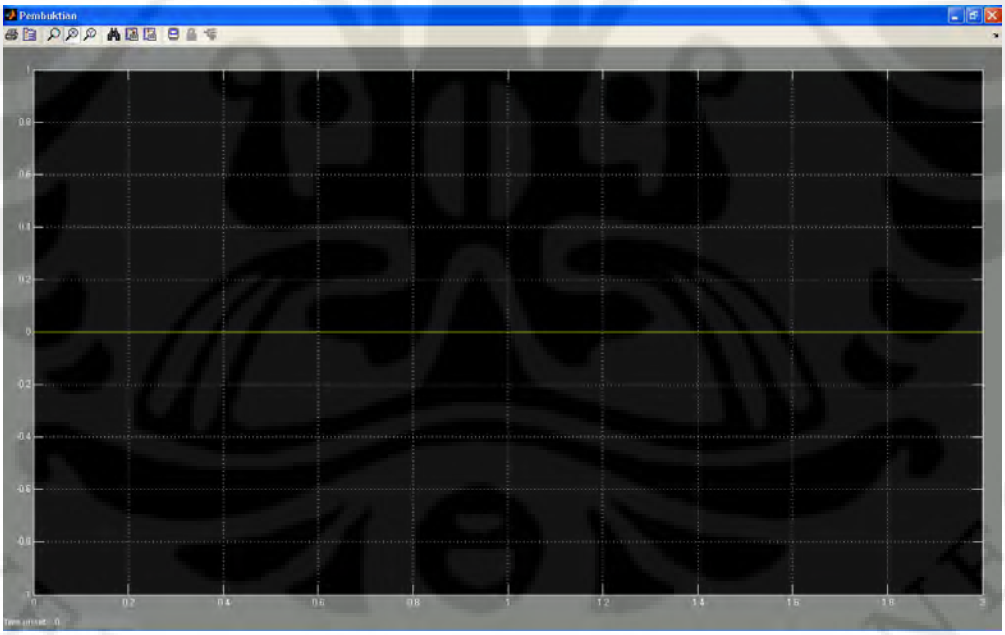
Perhitungan FFT dari variabel 'IFFT' seharusnya juga memberikan hasil yang sama dengan variabel 'modulasi'. Penghitungan FFT pada MATLAB untuk variabel IFFT memberikan hasil yang sangat mendekati dengan nilai dari variabel 'modulasi'. Memang terjadi perbedaan, namun sangat sedikit, pembulatan yang dilakukan akan memberikan hasil yang sama dengan variabel 'modulasi'.

Bentuk data dari variabel ini adalah vektor dengan ukuran [32x1] sesuai dengan ukuran dari masukan blok 'IFFT', sesuai dengan teori bahwa untuk *N point* FFT atau IFFT akan menghasilkan *N point* hasil transformasi.

Sinyal keluaran IFFT merupakan sinyal yang dikirim oleh *transmitter* OFDM. Sinyal tersebut diterima oleh *receiver* OFDM yang selanjutnya akan mengalami proses FFT dan demodulasi.

4.2.4 Pembuktian Data

Pengujian data yang telah diolah di *receiver* dilakukan dengan membandingkannya dengan data yang dikirim dari *transmitter*. Perbandingan data dilakukan dengan melakukan operasi *logical* XOR. Dengan membandingkan sinyal yang dikirim oleh *transmitter* dengan sinyal yang diterima di *receiver* yang telah diproses, dapat diketahui bahwa data yang diterima sama dengan yang dikirim seperti terlihat pada *scope* pembuktian pada Gambar 4.6.



Gambar 4.6 Perbandingan data

Grafik pada *scope* pembuktian selalu bernilai nol karena dilakukan proses XOR terhadap data yang dikirim dengan data yang diterima. Apabila data yang dikirim dengan data yang diterima bernilai sama maka hasil XOR sama dengan nol. Oleh karena data yang dikirim dengan data yang diterima selalu bernilai nol maka hasil

XOR juga akan selalu bernilai nol. Hal ini menunjukkan bahwa rancang bangun model rangkaian pengirim OFDM yang telah diuji sudah berfungsi dengan semestinya.

4.3. DSK TMS320C6713

Pada ujicoba dengan menggunakan DSK TMS320C6713, pengujian dilakukan dengan menggunakan *Real Time Data Exchange* (RTDX).

4.3.1 *Real Time Data Exchange* (RTDX)

Uji coba dilakukan dengan menggunakan RTDX karena *output* berupa angka. Apabila *output* yang berupa bit-bit dilihat dengan menggunakan *storage osciloscope* maka *output* yang terlihat akan mengalami gangguan disebabkan oleh adanya proses sampling sinyal *digital* menjadi sinyal *analog* sedangkan jika menggunakan RTDX maka *output* yang berupa bit-bit akan dikirim ke komputer dan data diolah secara *digital* sehingga tidak mengganggu atau merusak *output*.

Pengujian kebenaran data *output* dari *receiver* dilakukan dengan membandingkan data *output* RTDX dengan data *output* didapat dari Simulink. *Input* pada pengujian ini sama dengan *input* pada pengujian Simulink yakni *data.*

BAB 5

KESIMPULAN

Berdasarkan hasil analisis dapat disimpulkan :

1. Rangkaian pengirim OFDM dengan Huffman code dapat dibangun dengan menggunakan DSK TMS320C6713 dan telah berhasil diimplementasikan.
2. Pada implementasi rangkaian pengirim OFDM ini dengan menggunakan DSK TMS320C6713 masih terdapat gangguan berupa *noise*, tetapi pengaruh dari *noise* ini ternyata tidak signifikan sehingga tidak terlalu mengganggu.

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

[3] Budi Hartono, Bambang. Perancangan Dan Implementasi Huffman Coding Untuk Reduksi PAPR Pada Sistem OFDM. Program Studi Teknik Elektro UI. Depok

[4] http://en.wikipedia.org/wiki/Huffmancoding

[5] Langton, Charan., *"Orthogonal Frequency Division Multiplexing Tutorial"*, 2004.

[6] Matiae, Dusan. "*OFDM as a possible modulation technique for multimedia applications in the range of mm waves*", *Introduction to OFDM II Edition*.

[7] "Mengenal Teknologi *Frequency Division Multiplexing* (OFDM) pada Komunikasi *Wireless*", Elektro Indonesia, 5 Desember 2009. http://www.elektroindonesia.com/elektro/tel24.html

[8] "*Orthogonal Frequency Division Multiplexing*", OFDM *tutorial*. dari complextoreal. http://www.complextoreal.com/chapters/ofdm2.pdf

[9] Pietikäinen, Kari., *"Orthogonal Frequency Division Multiplexing"*, Postgraduate Course in Radio Communications, 2004.

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

[11] *Code Composer Studio Help*, *Texas Instrument*.

[12] Matlab *help, signal processing toolbox*, mathworks *inc.*

[13] *Introduction to Simulink*, *Link for CCS & Real-Time Workshop*, 16 November 2009. http://www.emba.uvm.edu/~mirchand/classes/EE275/2007/Real-Time/Lab6.pdf

LAMPIRAN

LAMPIRAN 1 ALGORITMA HUFFMAN CODING

```
%#########################
%Algoritma Huffman Coding
%#########################
function hufsig = fcn(sig)

eml.extrinsic('huffmandict', 'huffmanenco');
nsym=64;
n=5888;
nb=6;
bf=32;
dt=1000;

%---------------------------
%Tahap pembentukan Hystogram
%---------------------------
[r0 c0]=size(sig);
h=zeros(1,nsym);
y=sig';
s=0:(nsym-1);
w=zeros(1,n);
if any(sig)
    for i=1:r0
        h(double(y(i)+1))=h(double(y(i)+1))+1;
    end
end

%-------------------------------
%Tahap penghitungan probabilitas
%-------------------------------
prob=h/dt;

%-------------------------------
%Pembentukan pohon Huffman
%-------------------------------
[cl av]=huffmandict(s,prob);

%------------------------
%Pengkodean Huffman
%------------------------
w=huffmanenco(y,cl);
kr=nb-mod(n,nb);
ht=[w, zeros(1,kr)];
[b kr1]=size(ht);
o=kr1/6;
z=(6*((bf-mod(o,bf)) + o)) - kr1;
ht1=[ht, zeros(1,z)];

%-------------------------------
%Data output yang telah dikodekan
%-------------------------------
hufsig=ht1'
```