RANCANG BANGUN DEMODULATOR 16-QAM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASISKAN MATLAB SIMULINK

SKRIPSI

Oleh

AKHMAD ANDITO NEGORO

04 04 03 0059

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GENAP 2007/2008

RANCANG BANGUN DEMODULATOR 16-QAM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASISKAN MATLAB SIMULINK

SKRIPSI

Oleh

AKHMAD ANDITO NEGORO

04 04 03 0059

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GENAP 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul **:**

RANCANG BANGUN DEMODULATOR 16-QAM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASISKAN MATLAB SIMULINK

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 17 Juni 2008

(Akhmad Andito Negoro)

NPM 04 04 03 0059

PENGESAHAN

Skripsi dengan judul :

RANCANG BANGUN DEMODULATOR 16-QAM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASISKAN MATLAB SIMULINK

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia. Skripsi ini telah diujikan pada sidang ujian skripsi pada tanggal 29 Mei 2008 dan dinyatakan memenuhi syarat/sah sebagai skripsi pada Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.

Depok, 17 Juni 2008

Dosen Pembimbing

(Dr. Ir. Arman D. Diponegoro, M.Eng)
NIP. 131 476 472

UCAPAN TERIMA KASIH

Puji syukur hanya kepada Allah SWT Yang Maha Kuasa, sehingga skripsi ini dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada :

Dr. Ir. Arman D. Diponegoro, M.Eng

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Penulis juga mengucapkan terima kasih kepada orang tua dan adik penulis atas dukungannya dalam penyelesaian skripsi ini. Terima kasih juga penulis sampaikan kepada rekan-rekan mahasiswa/i Departemen Elektro Fakultas Teknik Universitas Indonesia khususnya angkatan 2004. Dan juga penulis mengucapkan terima kasih kepada berbagai pihak yang tidak dapat penulis sebutkan satu per satu atas bantuan dan dukungannya.

Akhmad Andito Negoro
NPM 04 04 03 005 9
Departemen Teknik Elektro

Dosen Pembimbing
Dr. Ir. Arman D. Diponegoro, M.Eng

RANCANG BANGUN DEMODULATOR 16-QAM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASISKAN MATLAB SIMULINK

ABSTRAK

Salah satu isu utama yang menjadi prioritas pada sebagian besar sistem telekomunikasi saat ini adalah efisiensi *bandwidth*. *Quadrature Amplitude Modulation* (QAM) hadir sebagai salah satu solusi dari permasalahan tersebut karena mampu menyediakan kapasitas informasi yang lebih tinggi. QAM menggunakan kombinasi perubahan amplitudo dan fasenya sekaligus. QAM menggunakan prinsip-prinsip dari teknik modulasi *Amplitude Modulation* dan *phase-shift (digital keying) modulation*.

Pada skripsi ini dirancang dan dibangun sebuah perangkat demodulator 16-QAM menggunakan MATLAB Simulink yang kemudian diimplementasikan pada DSK TMS320C6713 dengan bantuan fitur *Target for TIC6000*. Perancangan dengan Simulink ini berbasiskan *software* dan diagram blok sehingga dapat dihindari algoritma rumit yang mungkin ditemui pada bahasa pemrograman seperti bahasa C. Selain itu penggunaan Simulink menghindari perancangan secara *hardware.*

Rancang bangun demodulator disimulasikan pada Simulink terlebih dahulu dan selanjutnya diimplementasikan pada DSK TMS320C6713. Demodulator ini bertujuan untuk menerima data berupa sinyal termodulasi dari pengirim yang kemudian akan didemodulasikan kembali menjadi sinyal informasi yang sesuai dengan sinyal informasi dari sisi pengirim. Sinyal keluaran dari demodulator yang dibangun pada DSK disalurkan melalui *Real-Time Data Exchange* (RTDX).

Sinyal keluaran hasil demodulasi pada simulasi menggunakan Simulink memiliki bentuk yang sama dengan sinyal yang dikirim namun mengalami *delay*. Sinyal keluaran dari hasil ujicoba demodulator pada DSK TMS320C6713 juga menunjukkan hasil yang sama dengan simulasi menggunakan Simulink.

Kata Kunci : Demodulator 16-QAM, DSK TMS320C6713, Simulink, Matlab, RTDX

Akhmad Andito Negoro
NPM 04 04 03 005 9
Electrical Engineering Department

Counsellor
Dr. Ir. Arman D. Diponegoro, M.Eng

16-QAM DEMODULATOR DESIGN USING TMS320C6713 DSK BASED ON MATLAB SIMULINK

ABSTRACT

In most telecommunication systems, the high priority was on bandwidth efficiency. *Quadrature Amplitude Modulation* (QAM) was one of solutions to solve the problem, because it offered more information capacity, because QAM constructed in the combination of amplitude modulation and phase shift (digital keying) modulation.

The purpose of this minithesis was to design and build a 16-QAM demodulator using MATLAB Simulink, and then implemented it on DSK TMS320C6713 aided by *Target for TIC6000* fiture. The Simulink design was done based on the block diagram, therefore it could be avoid the complex algorithm that might be occurred in designing using C code. Utilizing the Simulink could be also to avoid the hardware-based design.

At first, the design of demodulator was simulated by means of Simulink and then it be implemented on DSK TMS320C6713. This demodulator had a purpose to receive modulated signal from transmitter, and then demodulate it back to information signal. The output signal from demodulator built on DSK was accessed via Real-Time Data Exchange (RTDX).

The output signal from simulation had a same form with the transmitted signal but it delayed. The output signal from demodulator built on DSK TMS320C6713 also indicate a same result with the simulation using Simulink.

Keywords: 16-QAM Demodulator, TMS320C6713 DSK, Simulink, Matlab, RTDX

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

ADC	*Analog to Digital Converter*
API	*Application Programming Interface*
ASK	*Amplitude Shift Keying*
CCS	*Code Composer Studio*
DAC	*Digital to Analog Converter*
DSK	*DSP starter kit*
DSP	*Digital Signal Processing*
DVB	*Digital Video Broadcast*
FIR	*Finite Impulse Response*
HDTV	*high-definition television*
IDE	*Integrated Development Environment*
JTAG	*Joint Team Action Group*
LO	*Local Oscillator*
PSK	*Phase Shift Keying*
QAM	*Quadrature Amplitude Modulation*
RTDX	*Real-Time Data Exchange*
TI	Texas Instrument
USB	*Universal Serial Bus*

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Isu utama pada sistem komunikasi saat ini dapat dikategorikan menjadi tiga yaitu: efisiensi *bandwidth*, efisiensi daya, dan efisiensi biaya. Efisiensi *bandwidth* menggambarkan kemampuan dari skema modulasi untuk menampung data pada *bandwidth* yang terbatas. Efisiensi daya menggambarkan kemampuan sistem untuk mengirimkan informasi pada *level* daya yang terendah yang dapat dilakukan. Masalah efisiensi *bandwidth* ini merupakan prioritas pada sebagian besar sistem telekomunikasi saat ini.

Teknik modulasi dijital menyediakan kapasitas informasi yang lebih tinggi, kompatibilitas yang lebih baik dengan "*digital data services"*, keamanan data yang lebih terjamin, kualitas komunikasi yang lebih baik, dan ketersediaan sistem yang lebih cepat [1].

Salah satu teknik modulasi dijital yang sering digunakan adalah *Quadrature Amplitude Modulation* (QAM). Berbagai protokol komunikasi saat ini telah mengimplementasikan QAM. Sebagai contohnya adalah protokol 802.11b *Wireless Ethernet* (Wi-Fi) dan *Digital Video Broadcast* (DVB) yang menggunakan modulasi 64-QAM. Selain itu, teknologi yang baru bermunculan saat ini seperti WiMAX dan HSDPA/HSUPA juga akan mengimplementasikan QAM [2].

Perangkat modulator atau demodulator QAM tersebut biasanya sudah terintegrasi di dalam sebuah *chip*. Selain dengan perancangan berbasis *hardware*, modulator atau demodulator QAM juga dapat dirancang dengan menggunakan *Digital Signal Processing* (DSP) yang berbasis *software* (program). Penggunaan DSP yang berbasis *software* ini tentunya lebih mudah dan sederhana dibandingkan dengan perancangan berbasis *hardware*.

Teknik DSP ini dapat diimplementasikan menggunakan *digital signal processors*. Prosesor yang banyak digunakan adalah prosesor-prosesor dari keluarga TMS320. Prosesor ini banyak digunakan pada telepon seluler, kamera dijital, *high-definition television* (HDTV), modem, dan divais-divais lainnya. Di

universitas biasanya digunakan prosesor TMS320C6x yang dibuat oleh *Texas Instruments*.

Prosesor DSP tersebut dapat diprogram dengan menggunakan bantuan MATLAB Simulink. Pemrograman dengan menggunakan Simulink tersebut cukup dilakukan dengan membuat rancangan model berbasiskan diagram blok yang dapat disimulasikan terlebih dahulu. Selanjutnya dari model tersebut, Simulink mampu meng-*generate* kode pemrograman yang akan diimplementasikan pada prosesor DSP.

1.2. TUJUAN PENULISAN

Tujuan dari penulisan skripsi ini adalah untuk merancang dan membangun sebuah demodulator 16-QAM pada DSP *starter kit* (DSK) TMS320C6713 dengan menggunakan bantuan Simulink.

1.3. BATASAN MASALAH

Masalah yang akan dibahas pada skripsi ini dibatasi pada rancang bangun demodulator 16-QAM menggunakan model Simulink tanpa menggunakan bahasa pemrograman. Hasil rancang bangun demodulator 16-QAM pada DSK tidak diujicobakan dengan *hardware* pada laboratorium (seperti *signal generator*) serta *logic analyzer*. Pada proses demodulasi 16-QAM ini tidak dibahas mengenai efisiensi *bandwidth*. Proses pengiriman sinyal dari modulator ke demodulator dilakukan pada satu papan DSK yang sama.

1.4. METODOLOGI PENELITIAN

Penelitian dilakukan dengan berdasarkan pada studi literatur, kemudian melakukan implementasi dengan membangun sebuah demodulator QAM, lalu melakukan evaluasi terhadap hasil unjuk kerja dari demodulator QAM yang telah dibangun.

1.5. SISTEMATIKA PENULISAN

Laporan ini terdiri atas lima (5) bab. Masing-masing bab akan dipaparkan sebagai berikut :

i) BAB 1 PENDAHULUAN

Bab ini menjelaskan Latar Belakang, Tujuan Penulisan, Pembatasan Masalah, Metodologi Penulisan dan Sistematika Penulisan.

ii) BAB 2 TINJAUAN PUSTAKA

Bab ini menjelaskan dasar teori dari teknik QAM, Simulink, dan DSK TMS320C6713.

iii) BAB 3 RANCANG BANGUN

Bab ini membahas tentang perancangan model demodulator 16-QAM pada Simulink serta perancangan demodulator 16-QAM untuk diimplementasikan pada DSK TMS320C6713.

iv) BAB 4 UJI COBA DAN ANALISIS

Bab ini membahas mengenai hasil pengujian serta analisis dari sistem demodulator 16-QAM yang dijalankan pada DSK TMS320C6713.

v) BAB 5 KESIMPULAN

Bab ini berisi poin-poin kesimpulan yang didapat dari hasil analisis dan uji coba.

BAB II
QAM, SIMULINK, DSK TMS320C6713, DAN *LINK FOR CCS*

2.1 *QUADRATURE AMPLITUDE MODULATION* (QAM)

2.1.1 Pengertian Modulasi dan QAM

Modulasi adalah suatu proses penumpangan sinyal yang hendak dikirim pada sebuah sinyal pembawa (*carrier*). Sinyal data dapat ditumpangkan ke sinyal pembawa dengan cara mengubah amplitudo, frekuensi, atau fase dari sinyal pembawa tersebut. Untuk meningkatkan kapasitas informasi yang dikirimkan, dapat dilakukan dengan perubahan terhadap kombinasi 2 dari beberapa parameter tersebut [3]. Metode tersebut digunakan pada skema modulasi QAM.

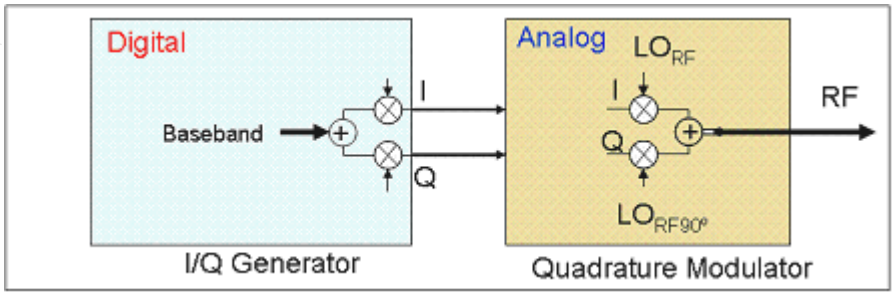
QAM adalah skema modulasi dimana data ditumpangkan pada gelombang pembawa dengan cara mengubah amplitudo dari 2 buah gelombang pembawa tersebut. Dua buah gelombang pembawa ini biasanya adalah gelombang sinusoid dan saling berbeda fase 90° satu sama lain, sehingga disebut *quadrature carriers* [4].

2.1.2 Konsep Modulasi QAM

QAM merupakan salah satu teknik modulasi dijital multisimbol. Pada QAM, setiap simbol yang dikirimkan melambangkan beberapa bit sekaligus. Oleh karena itu, QAM dapat digunakan untuk mendapatkan *data rate* yang tinggi pada *bandwidth* yang terbatas. Pada QAM, fase dan amplitudo dari sinyal pembawa yang diubah-ubah [3]. Cara pengubahan gelombang pembawa ini merupakan perpaduan dari 2 teknik modulasi yaitu *Phase Shift Keying (PSK)* dan *Amplitude Shift Keying (ASK)*.

Proses modulasi QAM secara umum dapat ditunjukkan pada Gambar 2.1. Pada gambar tersebut, sinyal informasi yang akan dikirim (*baseband*) dibagi menjadi 2 komponen yaitu komponen I dan Q yang saling berbeda fase sebesar 90°. Komponen I adalah komponen "*in*-phase" yang merupakan bagian riil dari

sinyal modulasi QAM dan komponen Q adalah komponen "*quadrature*" yang merupakan bagian imajiner dari sinyal modulasi.



Gambar 2.1 Diagram blok proses modulasi QAM [2]

Kemudian, sinyal I dan Q tersebut dikalikan gelombang yang berasal dari LO (*Local Oscillator*) dimana fase dari gelombang yang dihasilkan kedua LO tersebut juga saling berbeda 90° [2]. Setelah itu, sinyal I dan Q yang telah dicampur dengan gelombang pembawa tersebut dijumlahkan menjadi sebuah sinyal yang dapat ditransmisikan.

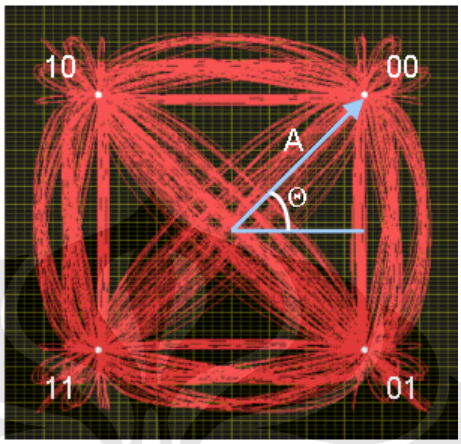
Dari penjelasan proses modulasi QAM diatas, dapat dituliskan persamaan dari sinyal QAM tersebut seperti pada Persamaan (2.1) [4]. Pada persamaan tersebut, sinyal I dan Q masing-masing dikalikan dengan gelombang cosinus dan sinus yang berfrekuensi sama (frekuensi *carrier* f_c) sebelum keduanya dijumlahkan menjadi sinyal termodulasi *s(t)*.

$$s(t) = I(t)\cos(2\pi f_c t) + Q(t)\sin(2\pi f_c t) \quad \text{.....(2.1)}$$

Seperti yang telah disebutkan sebelumnya bahwa modulasi QAM mengirimkan informasi dijital dengan cara mengubah-ubah fase dan amplitudo dari gelombang elektromagnetik sinusoidal secara periodik. Sebagai contoh, modulasi 4-QAM menggunakan sinyal dengan 4 macam kombinasi unik antara fase dengan amplitudonya. Setiap kombinasi tersebut disebut simbol dimana pada 4-QAM setiap simbolnya memiliki pola dijital 2-bit. Misalnya *bitstream* yang dibuat adalah (1,0,0,1,1,1), maka bit-bit tersebut dikelompokkan setiap 2 bit menjadi (10,01,11), sehingga *bitstream* tersebut dapat dipetakan pada simbol yang sesuai untuk 4-QAM [2].

2.1.3 Diagram Konstelasi

Amplitudo dan fase untuk masing-masing simbol pada QAM dapat digambarkan dalam sebuah diagram 2 dimensi yang disebut sebagai diagram konstelasi. Pada Gambar 2.2 ditunjukkan diagram konstelasi untuk modulasi 4-QAM.



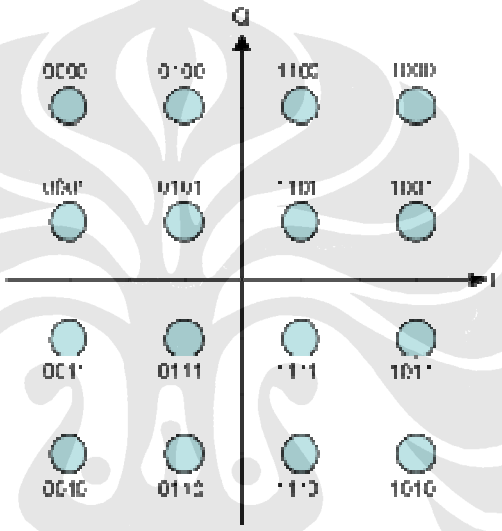
Gambar 2.2 Diagram konstelasi 4-QAM [2]

Seperti dijelaskan sebelumnya, bahwa pada modulasi 4-QAM digunakan 4 buah simbol yang masing-masing dapat direpresentasikan dengan fase (Θ) dan amplitudo (A) yang unik. Pada Gambar 2.2, masing-masing simbol tersebut ditunjukkan berupa titik putih. Garis merah menunjukkan transisi fase dan amplitudo dari satu simbol ke simbol lainnya. Pola bit dijital yang direpresentasikan oleh masing-masing simbol (00, 10, 11, dan 01) juga ditamplikan pada gambar tersebut [2].

Modulasi 4-QAM memungkinkan pengiriman sebanyak 2 bit per simbol. *Data rate* yang lebih tinggi dapat diperoleh dengan meningkatkan jumlah simbol pada diagram konstelasi. Menurut perjanjian, jumlah simbol pada peta simbol (diagram konstelasi) disebut "M" dan merupakan "M-ary" pada skema modulasi. Atau dengan kata lain, 4-QAM memiliki M-ary bernilai 4 dan 256-QAM memiliki 256 M-ary. Jumlah bit yang dapat direpresentasikan oleh sebuah simbol memiliki hubungan logaritmik dengan M-ary. Sebagai contoh, 2 bit dapat direpresentasikan oleh setiap simbol pada 4-QAM, sedangkan data 4 bit dengan 16-QAM. Hubungan tersebut dapat dinyatakan dengan Persamaan (2.2) [2].

$$\text{Jumlah bit per simbol} = \log_2(M) \quad \text{..... (2.2)}$$

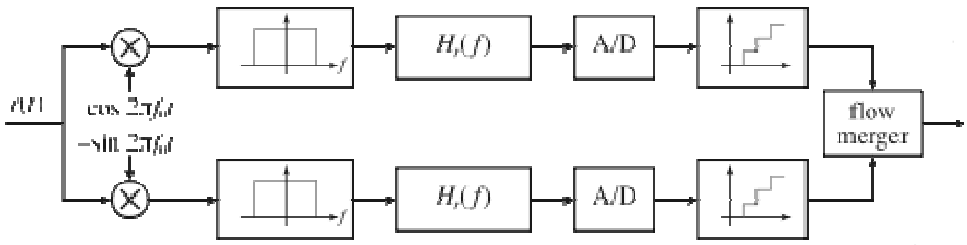
Diagram konstelasi untuk modulasi 16-QAM ditunjukkan pada Gambar 2.3. Pada gambar tersebut ada 16 buah titik yang berarti ada 16 simbol yang digunakan. Pada setiap titik di gambar tersebut juga ditunjukkan pola bit dijital yang sesuai dengan perjanjian *Gray Code*. Sumbu horisontal merupakan sumbu yang mewakili komponen cos $\omega_c t$ dan disebut sebagai sumbu I (*inphase*), sedangkan sumbu vertikal adalah sumbu yang mewakili komponen sin $\omega_c t$ dan disebut sebagai sumbu Q (*quadrature*) [3]. Konfigurasi titik pada diagram konstelasi tidak harus seperti pada gambar. Gambar 2.3 hanyalah mewakili konfigurasi titik untuk "*rectangular* QAM".



Gambar 2.3 Diagram konstelasi 16-QAM [4]

2.1.4 Demodulasi QAM

Pada sisi penerima (*receiver*), sinyal QAM yang diterima harus didemodulasikan agar dapat diperoleh kembali sinyal informasi sesuai dengan yang dikirimkan dari pengirim (*transmitter*). Proses demodulasi tersebut dapat digambarkan seperti pada Gambar 2.4.



Gambar 2.4 Diagram blok demodulator QAM [4]

Proses demodulasi ini dapat dilakukan dengan cara mengalikan sinyal yang diterima dengan gelombang kosinus, yaitu seperti pada Persamaan (2.5) [4].

$$\begin{aligned} r_i(t) &= s(t)\cos(2\pi f_c t) \\ &= I(t)\cos(2\pi f_c t)\cos(2\pi f_c t) + Q(t)\sin(2\pi f_c t)\cos(2\pi f_c t) \end{aligned} \quad (2.5)$$

dengan menggunakan identitas trigonometri, maka Persamaan (2.5) di atas dapat diubah menjadi Persamaan (2.6) [4].

$$\begin{aligned} r_i(t) &= \frac{1}{2} I(t)\left[1+\cos(4\pi f_c t)\right] + \frac{1}{2} Q(t)\sin(4\pi f_c t) \\ &= \frac{1}{2} I(t) + \frac{1}{2}\left[I(t)\cos(4\pi f_c t) + Q(t)\sin(4\pi f_c t)\right] \end{aligned} \quad \text{.....(2.6)}$$

Sinyal yang ditunjukkan oleh Persamaan (2.6) mengandung sinyal yang berfrekuensi tinggi, yaitu komponen dengan frekuensi $4\pi f_0 t$. Dengan melewatkan sinyal tersebut melalui *low pass filter* (H_r), maka sinyal berfrekuensi tinggi (yang mengandung $4\pi f_c t$) akan dihilangkan, sehingga diperoleh sinyal I(t) yang merupakan komponen *in-phase* dari sinyal yang dikirimkan.

Hal yang serupa juga dilakukan untuk memperoleh sinyal Q(t), yaitu dengan mengalikan sinyal s(t) dengan gelombang sinus. Setelah diperoleh sinyal I dan Q, maka selanjutnya kedua sinyal tersebut dikonversi dan digabungkan kembali dengan *"flow merger"* agar kembali lagi ke dalam bentuk yang sama dengan sinyal informasi yang dikirim.

2.2 SIMULINK

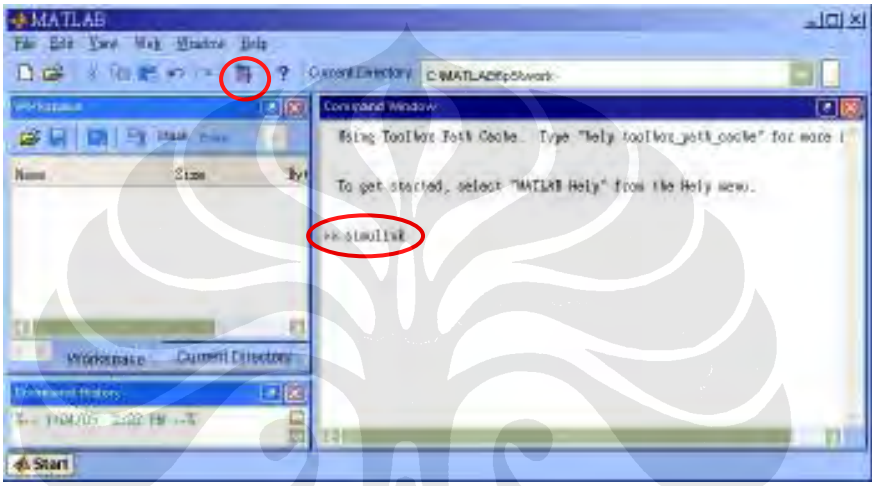
Komputer saat ini telah mempu melakukan perhitungan matematika yang sangat rumit sekalipun. Hal ini dapat digunakan untuk mensimulasikan (atau meniru) sistem *dynamic* tanpa harus terlebih dahulu membuat sistem fisik yang sebenarnya. Dengan dilakukannya simulasi dapat menghemat waktu dan biaya yang dapat dihabiskan untuk pembuatan *prototype* sistem tersebut [5].

Simulink yang merupakan singkatan dari *"**Simu**lation and **Link**"* adalah salah satu program tambahan dari MATLAB yang dibuat oleh Mathworks Inc., sebuah perusahaan yang berbasis di Natick, MA. Simulink dapat melakukan pemodelan, simulasi, dan analisis suatu sistem dengan bantuan graphical user interface (GUI) [6].

MATLAB memiliki banyak versi dimana pada setiap versi memiliki *library* yang berbeda-beda. MATLAB yang digunakan pada skripsi ini adalah versi 7.4.0.287 (R2007a).

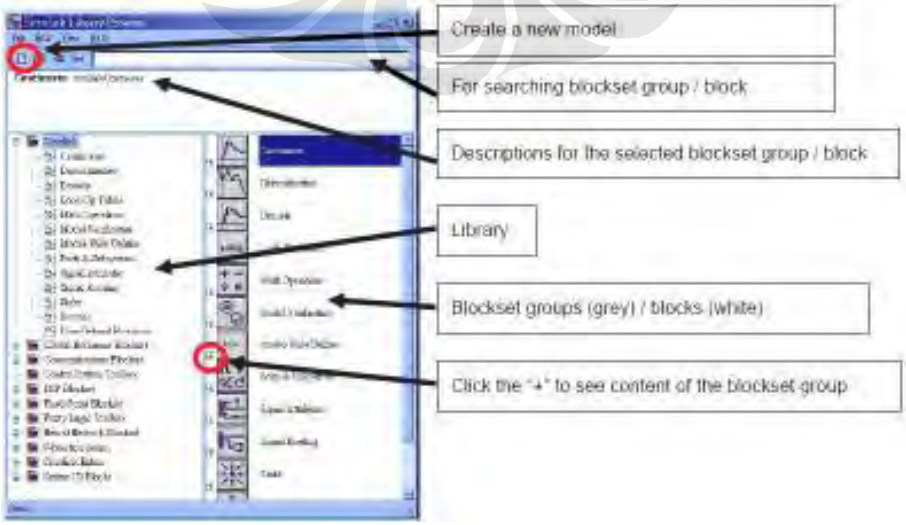
2.2.1 Menjalankan Simulink

Untuk memulai Simulink dapat dilakukan dengan menjalankan MATLAB terlebih dahulu dan mengetik "simulink" pada *command window*, atau mengklik *icon* pada toolbar seperti ditunjukkan pada Gambar 2.5.



Gambar 2.5 Menjalankan Simulink dari MATLAB

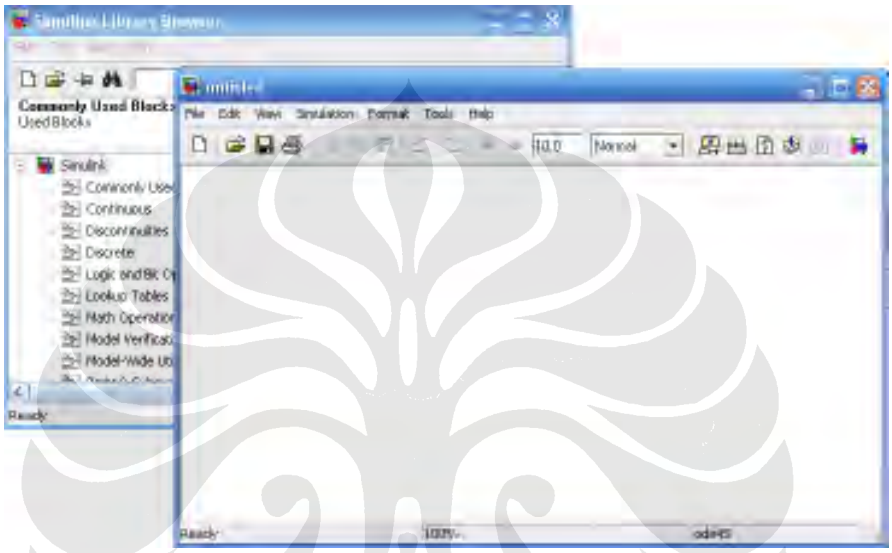
Setelah itu, akan muncul *"Simulink Library Browser"* pada *window* baru seperti ditunjukkan pada Gambar 2.6. Simulink menyediakan banyak sekali *blockset* yang dapat digunakan untuk membuat pemodelan.



Gambar 2.6 Simulink *Library Browser* [6]

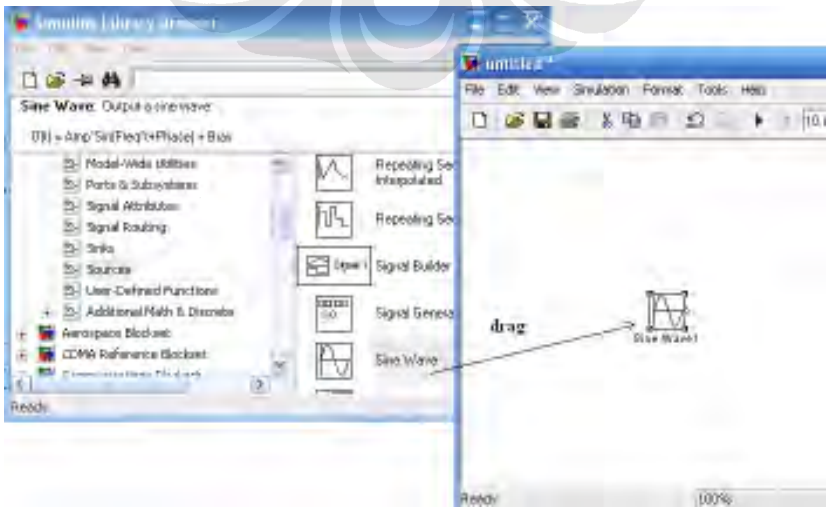
2.2.2 Pemodelan pada Simulink

Untuk membuat model pertama-tama dapat dilakukan dengan mengakses menu *File > New > Model* pada *"Simulink Library Browser"*, atau dengan cara meng-klik ikon *"Create a new model"*. Selanjutnya akan muncul *window* baru dimana blok-blok untuk pembuatan model dapat diletakkan. Pembuatan model baru tersebut ditunjukkan pada Gambar 2.7.



Gambar 2.7 Memulai model baru pada Simulink

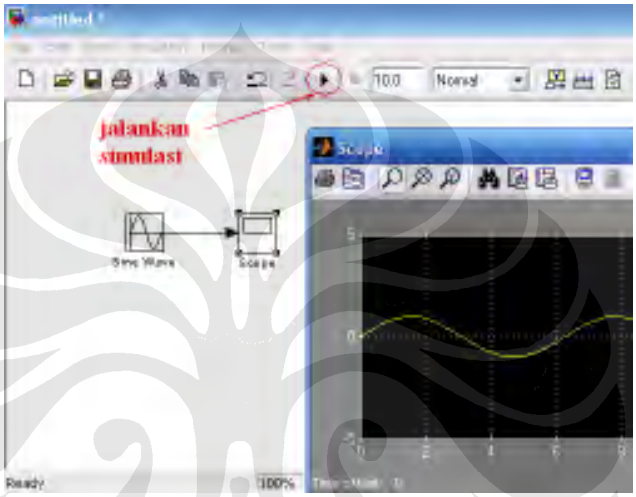
Untuk meletakkan blok ke dalam model dapat dilakukan dengan men-*drag* blok tersebut dari *"Simulink Library Browser"* ke model yang dibuat seperti ditunjukkan pada Gambar 2.8.



Gambar 2.8 Memasukkan blok ke dalam model

Dengan men-*doubleclick* blok pada model, kita dapat mengakses *window "Block Parameter"* untuk blok tersebut. Dari *"Block Parameter"* ini dapat diatur parameter-parameter dari setiap blok.

Blok-blok yang ada pada model dapat dihubungkan dengan cara men-*drag* tanda panah dari blok asal ke blok tujuan, atau dengan menekan tombol "Ctrl" kemudian sambil mengklik blok asal lalu blok tujuannya. Model yang sudah dibuat dapat dijalankan simulasinya dari Simulink dengan cara meng-klik ikon *"run"* pada *toolbar* seperti ditunjukkan pada Gambar 2.9.



Gambar 2.9 Menghubungkan blok dan menjalankan simulasi

2.2.3 *Target for TI C6000*

Target for TI C6000 merupakan fitur dari MATLAB yang mampu mengintegrasikan Simulink dan MATLAB dengan DSP *tools*. Dengan *Target for TI C6000*, maka perancangan algoritma DSP dapat dilakukan pada Simulink dan kemudian *Real-Time Workshop* akan membangkitkan kode C yang ditargetkan pada papan DSP atau *Code Composer Studio Integrated Development Environment* (CCS IDE).

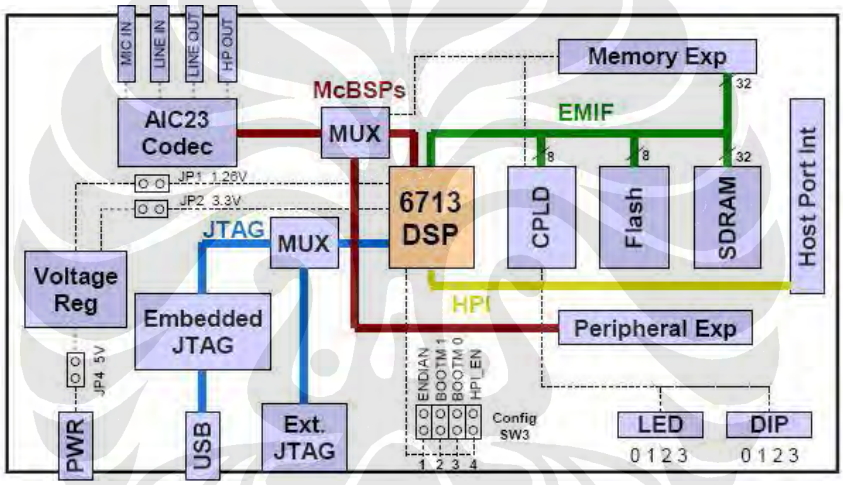
Target for TI C6000 menggunakan kode C yang telah dibangkitkan sebelumnya dan menggunakan DSP *tools* untuk membangun kode mesin yang sesuai dengan papan DSP yang digunakan. Proses tersebut dilakukan dengan mendownload kode mesin pada papan DSP dan menjalankannya pada prosesor DSP. Setelah proses *download* kode selesai dilakukan, aplikasi DSP yang dirancang akan berjalan secara otomatis pada target. *Hardware* yang didukung

oleh MATLAB diantaranya adalah DSK C6713 yang akan digunakan pada skripsi ini.

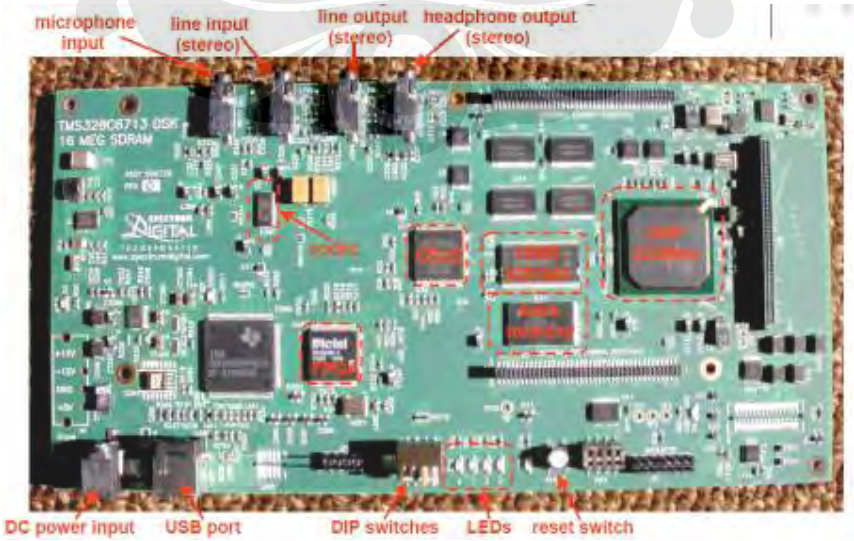
2.3 DSP STARTER KIT (DSK) TMS320C6713

DSP keluarga TMS320C6x (C6x) merupakan mikroprosesor berkecepatan tinggi dengan arsitektur dan set instruksi yang khusus dan cocok untuk pemrosesan sinyal. Notasi C6x digunakan untuk menandakan anggota keluarga prosesor TMS320C6000 dari Texas Instrument (TI). Arsitektur prosesor C6x sangat sesuai untuk perhitungan numerik yang sulit dan kompleks [7].

Blok diagram dari DSK TMS320C6713 ditunjukkan pada Gambar 2.10 dan gambar *layout* dari papan DSK ditunjukkan pada Gambar 2.11.



Gambar 2.10 Blok diagram DSK TMS320C6713 [9]



Gambar 2.11 *Layout* dari papan DSK TMS320C6713 [8]

Fitur-fitur utama dari DSK TMS320C6713 adalah sebagai berikut:

a) TMS320C6713 beroperasi pada 225 MHz
b) AIC23 stereo codec (ADC dan DAC)
c) 16 MB synchronous DRAM
d) 512 KB non-volatile Flash memory
e) 4 LED dan 4 DIP switch
f) *Universal Serial Bus* (USB) interface to PC
g) Configurable boot options
h) Konektor untuk *expansion card*
i) Single voltage power supply (+5V)

Papan DSK memiliki 4 buah konektor yang menyediakan fasilitas untuk *input* dan *output*. Konektor-konektor tersebut yaitu: MIC IN, LINE IN, LINE OUT, dan HEADPHONE. Keempat konektor tersebut melibatkan *onboard codec* AIC23 yang menyediakan *Analog to Digital Converter* (ADC) dan *Digital to Analog Converter* (DAC) dimana konektor masukan (MIC IN dan LINE IN) berhubungan dengan ADC dan konektor keluaran (LINE OUT dan HEADPHONE) berhubungan dengan DAC. *Codec* ini bekerja pada *clock frequency* 12 MHz [7].

Analisis secara *real-time* dapat dilakukan dengan menggunakan *Real-Time Data Exchange* (RTDX). RTDX memungkinkan pertukaran data diantara komputer dan target DSK. Komunikasi antara komputer dan DSK ini dapat terjadi melalui modul *Joint Team Action Group* (JTAG) yang ber-*interface* melalui *port* USB.

2.4 CODE COMPOSER STUDIO (CCS) DAN LINK FOR CCS

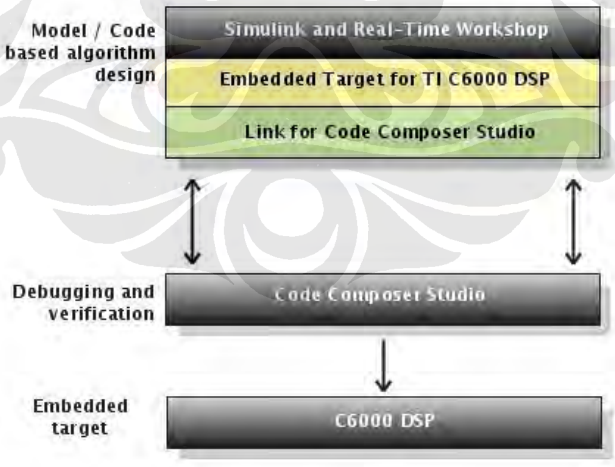
CSS menyediakan sebuah *integrated development environment* (IDE) yang dapat membangkitkan kode program untuk dijalankan pada DSK. Beberapa *tool* untuk pembangkitan kode yang terdapat di dalam CCS yaitu C *compiler*, *assembler* dan *linker*. CCS memiliki dukungan grafis dan *real-time debugging*.

C *compiler* berfungsi untuk mengkompilasi program berbahasa C dengan ekstensi .c sehingga dihasilkan program berbahasa *assembly* dengan ekstensi .asm. *Assembler* berfungsi untuk mengubah file berekstensi .asm tadi menjadi file

dengan bahasa mesin berekstensi .obj. *Linker* berfungsi untuk memadukan *object files* dan *object library* sebagai input untuk menghasilkan file yang dapat dieksekusi berekstensi .out. File *executable* ini dapat dibebankan dan dijalankan langsung pada DSK C6713 [7].

Seperti telah disebutkan di atas bahwa CCS menggunakan bahasa pemrograman C dan *assembly*. Program dalam bahasa C tersebut dapat menjadi rumit terlebih lagi untuk desain tingkat tinggi. Penggunaan Simulink dalam perancangan dapat menghindari kerumitan dalam pembuatan algoritma pemrograman tersebut. Hal ini karena pemrograman Simulink berbasiskan pada blok diagram [9].

Real-Time Workshop mengubah model Simulink kedalam kode ANSI C/C++ yang dapat dikompilasi menggunakan CCS. Embedded Target for TI C6000 DSP menyediakan *Application Programming Interface* (API) yang dibutuhkan oleh Real Time Workshop untuk membangkitkan kode yang spesifik untuk platform C6000. Link for CCS digunakan untuk proses *code building*. Kode ini kemudian dapat didownload pada target. Data pada target dapat diakses dengan CCS atau Matlab melalui *link for CCS* atau melalui RTDX seperti ditunjukkan pada Gambar 2.12.



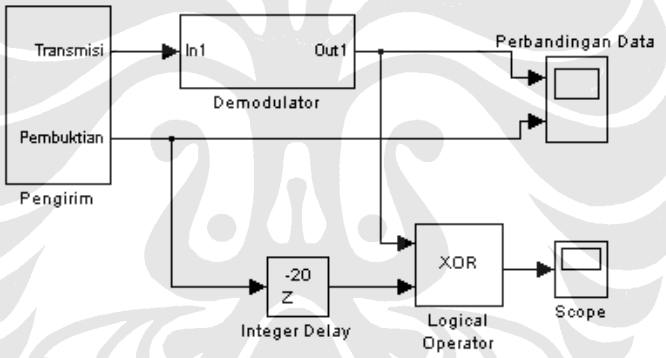
Gambar 2.12 Diagram alir yang menggambarkan hubungan antara Simulink dan *real-time workshop* dengan DSP C6000 [9]

BAB III
RANCANG BANGUN

3.1 RANCANG BANGUN SISTEM MENGGUNAKAN SIMULINK

3.1.1 Rancang Bangun Keseluruhan Sistem

Diagram blok sistem yang dibuat pada Simulink secara keseluruhan diperlihatkan pada Gambar 3.1. Secara garis besar proses yang terjadi dimulai dari pengiriman sinyal dari pengirim, lalu sinyal tersebut diterima dan didemodulasikan kembali oleh blok demodulator 16-QAM. Sinyal yang telah diperoleh dari hasil demodulasi kemudian dilihat melalui *scope* "perbandingan data" dan dibandingkan dengan sinyal yang dikirim. Penjelasan dari beberapa blok pada Gambar 3.1 akan dijelaskan pada bagian berikutnya.



Gambar 3.1 Rancang bangun sistem secara keseluruhan

3.1.1.1 *Logical Operator XOR*

Blok ini berfungsi untuk melakukan operasi logika XOR dari setiap masukkannya. Masukan untuk blok ini dianggap bernilai BENAR/*TRUE* (1) jika nilainya bukan nol dan bernilai SALAH/*FALSE* (0) jika nilainya nol. Operasi logika XOR akan menghasilkan keluaran BENAR jika masukan yang bernilai BENAR ada n masukan dimana n adalah bilangan ganjil.

Pada rancangan, blok ini berfungsi untuk melakukan perbandingan antara sinyal yang dikirim dengan yang diterima. Pada rancangan ini, masukannya ada 2 buah sehingga jika kedua masukan tersebut memiliki nilai yang sama, maka

keluarannya akan bernilai 0. Namun, jika hanya ada sebuah masukan yang bernilai 1 maka keluarannya akan bernilai 1.

3.1.1.2 *Integer Delay*

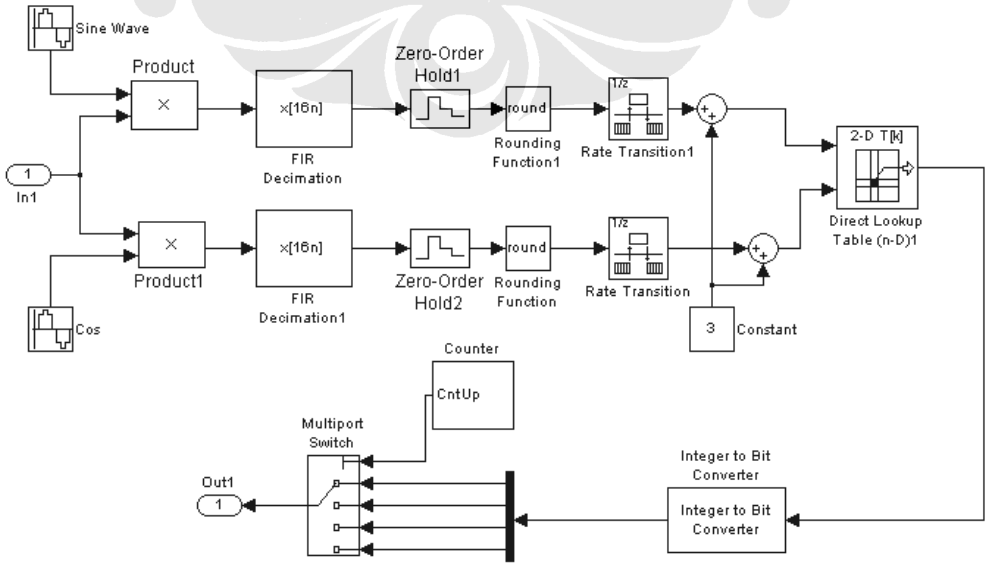
Blok ini berfungsi untuk memberikan *delay* pada sinyal selama periode N sampel. Pada rancangan, blok ini memberi *delay* pada sinyal masukan sebanyak 20 *sample period*. Hal ini dilakukan agar sinyal pada pengirim dapat dibandingkan dengan sinyal pada penerima yang mengalami *delay*.

3.1.1.3 *Scope*

Blok ini befungsi untuk menampilkan bentuk sinyal pada simulasi. Pada rancangan, sinyal yang ditampilkan adalah sinyal yang dikirim dengan sinyal yang diterima.

3.1.2 Rancang Bangun Blok Demodulator

Rancang bangun demodulator secara spesifik ditunjukkan pada Gambar 3.2. Blok-blok yang digunakan untuk membangun demodulator yaitu: *Sine wave, Product, Lowpass Filter, Finite Impulse Response (FIR) decimation, Zero-Order Hold, Rounding Function, Rate Transition, Constant, Sum, Direct Lookup Table, Integer to Bit Converter, Demux, Counter,* dan *Multiport Switch*. Fungsi dari masing-masing blok tersebut akan dibahas pada bagian berikutnya.

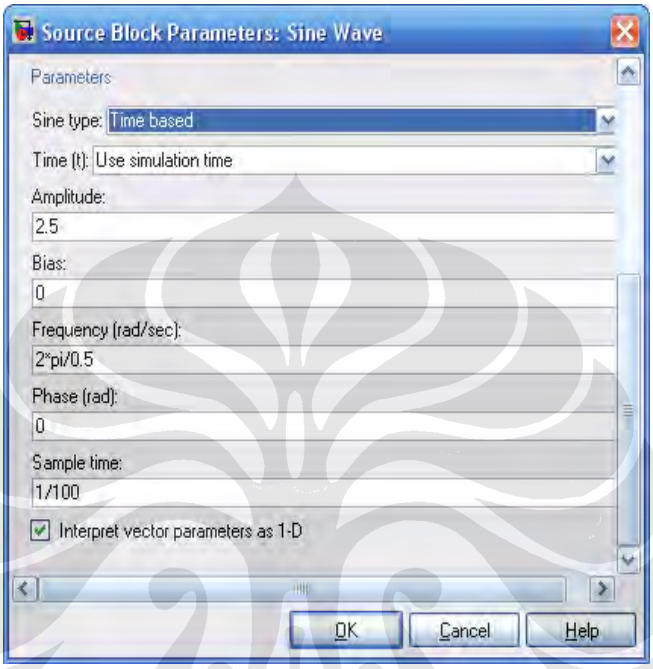


Gambar 3.2 Rancang bangun blok demodulator

3.1.2.1 *Sine Wave*

Blok ini berfungsi untuk membangkitkan gelombang sinusoid. Pada rancangan ini digunakan gelombang sinus dengan mode *time based*. Keluaran dari blok ini ditentukan oleh Persamaan (3.1).

$$y = Amplitude \times \sin\left(frequency \times time + phase \right) + bias \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$



Gambar 3.3 Pengaturan parameter blok *sine wave*

Pengaturan parameter untuk blok *sine wave* ini diperlihatkan pada Gambar 3.3. Secara *default*, nilai *sample time* adalah 0 yang menghasilkan gelombang sinus dengan mode kontinu. Namun dengan memberikan nilai *sample time* sebesar 1/100, maka gelombang yang dihasilkan akan berbentuk diskrit. Blok *Cos* yang digunakan untuk membangkitkan gelombang kosinus menggunakan parameter yang sama dengan blok *sine wave*, hanya saja pada parameter *Phase* diatur sebesar pi/2.

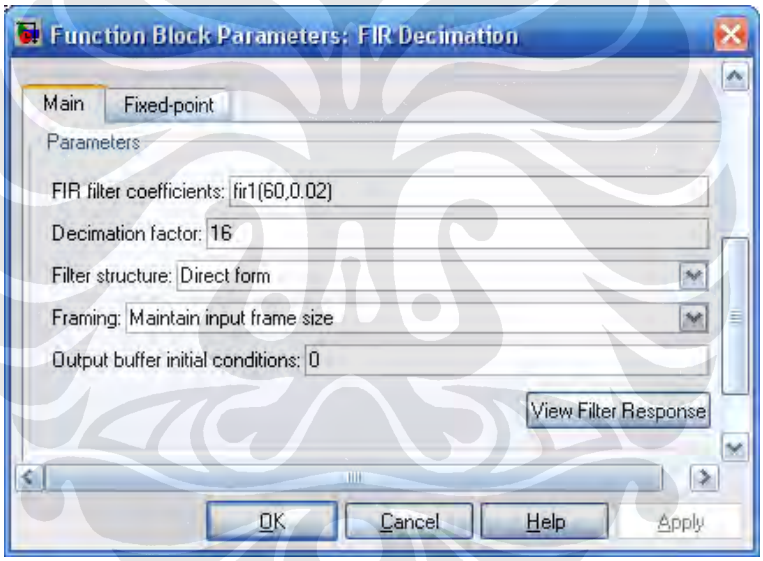
3.1.2.2 *Product*

Blok ini berfungsi untuk mengalikan dua buah gelombang masukan. Pada rancangan ini blok *product* akan mengalikan sinyal termodulasi dengan gelombang sinus dan kosinus.

3.1.2.3 *FIR Decimation*

Blok ini berfungsi untuk melakukan fungsi filter dan *downsample* (menurunkan *sample rate*) dari sinyal masukan. Pada parameter *FIR Filter coefficients* diberikan nilai fir1(60,0.02). Fungsi fir1(60,0.02) ini akan menghasilkan koefisien-koefisien untuk filter FIR *lowpass* dengan orde 60 dan frekuensi *cutoff* ternormalisasi sebesar 0,02. Dengan demikian blok ini akan berfungsi sebagai *lowpass filter* yang akan melewatkan gelombang dengan frekuensi rendah dan memblok frekuensi tinggi.

Selanjutnya parameter *Decimation factor* diatur sebesar 16 yang berarti *sample rate* dari masukan akan diturunkan 16 kalinya sehingga keluarannya memiliki *sample rate* yang lebih rendah. Pengaturan parameter untuk blok *FIR Decimation* ini diperlihatkan pada Gambar 3.4.



Gambar 3.4 Pengaturan parameter blok *FIR Decimation*

3.1.2.4 *Zero-Order Hold*

Blok ini berfungsi untuk melakukan *sampling* dan menahan sinyal masukan selama *Sample time* yang ditentukan. Parameter *Sample time* diatur sebesar 4. Dengan demikian keluaran dari blok ini adalah berupa sinyal diskrit dengan *sample time* sebesar 4.

3.1.2.5 *Rounding Function*

Blok ini berfungsi untuk melakukan pembulatan terhadap sinyal masukan. Metoda pembulatan yang digunakan adalah mode *round*. Mode *round* akan membulatkan sinyal masukan ke nilai integer yang terdekat.

3.1.2.6 *Rate Transition*

Blok ini melakukan transfer data dari keluaran suatu blok yang beroperasi pada *rate* tertentu menuju masukan pada blok lain yang beroperasi pada *rate* yang berbeda. Pada rancangan ini, blok tujuan yaitu blok *Sum* beroperasi pada *sample time* sebesar 1, sedangkan blok asal yaitu blok *round* beroperasi pada *sample time* sebesar 4 sehingga blok *Rate Transition* bekerja sebagai *unit delay* yang akan mempercepat *sample time* dari 4 menjadi 1.

3.1.2.7 *Constant*

Blok ini berfungsi untuk menghasilkan nilai konstanta sesuai keiinginan. Parameter *Constant value* pada blok ini diatur senilai 3 dan parameter *Sample time* diatur *inf*. Pengaturan tersebut akan menyebabkan keluaran dari blok ini terus menerus bernilai 3. Nilai yang diberikan adalah 3 karena dalam rancangan demodulator 16-QAM menggunakan diagram konstelasi yang memiliki nilai amplitude minimum -3.

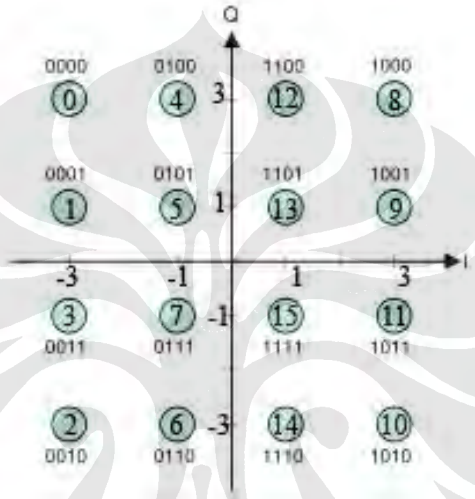
3.1.2.8 *Sum*

Blok ini berfungsi untuk menjumlahkan sinyal-sinyal masukan. Pada rancangan ini, blok *Sum* akan menjumlahkan sinyal hasil pembulatan dari blok *Round* dengan nilai konstan 3. Hal ini bertujuan agar nilai masukan untuk blok *Direct Lookup Table* bernilai minimal 0, karena prinsip *indexing* yang digunakan berbasis 0. Nilai keluaran dari blok *Round* adalah sinyal dengan nilai yang spesifik yaitu : -3, -1, 1, dan 3 sehingga setelah dijumlahkan dengan nilai konstan 3 akan menjadi sinyal dengan nilai 0, 2, 4, dan 6.

3.1.2.9 *Direct Lookup Table*

Tabel yang digunakan adalah tabel berdimensi 2. Blok ini menggunakan masukan-masukannya sebagai indeks berbasis nol yang digunakan untuk menunjuk elemen pada tabel *lookup*.

Pada rancangan ini, tabel *lookup* berfungsi untuk memetakan kembali sinyal-sinyal *inphase (I)* dan *quadrature (Q)* ke dalam simbol integer yang bersesuaian. Skema pemetaan ini berdasarkan pada diagram konstelasi yang ditunjukkan pada Gambar 3.5.



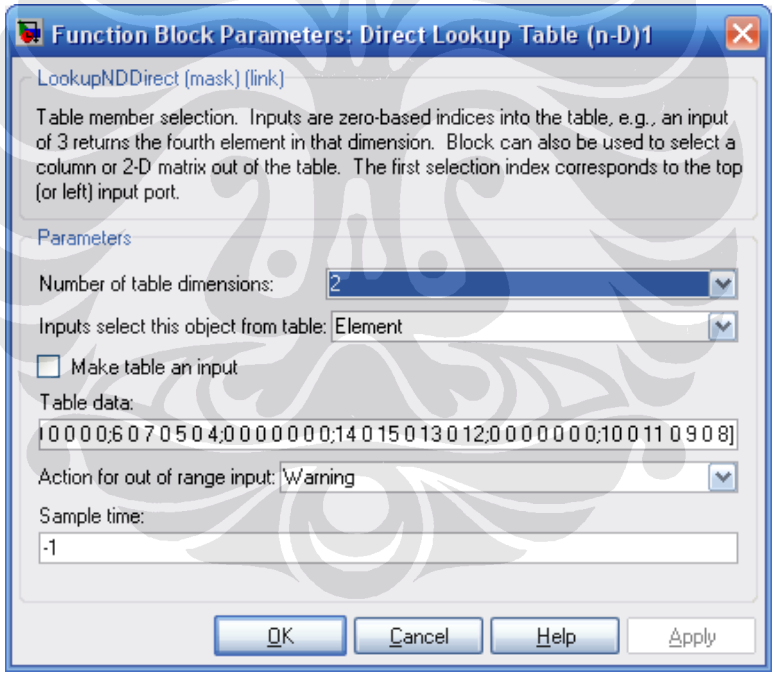
Gambar 3.5 Diagram konstelasi

Seperti telah dijelaskan sebelumnya bahwa sinyal I dan Q akan mempunyai nilai yang spesifik yaitu : -3, -1, 1, dan 3 dan setelah dijumlahkan dengan 3, maka sinyal yang masuk ke blok *Direct Lookup Table* memiliki nilai 0, 2, 4, dan 6. Berdasarkan diagram konstelasi dan nilai masukan tersebut, maka dapat dibuat tabel *lookup* seperti ditunjukkan pada Tabel 3.1.

Dapat dilihat pada Tabel 3.1 bahwa indeks 1, 3, dan 5 tidak digunakan sehingga nilai elemen pada tabel untuk indeks-indeks tersebut adalah nol. Nilai elemen-elemen dari tabel ditulis pada parameter *table data* dalam bentuk *array* yaitu : [2 0 3 0 1 0 0;0 0 0 0 0 0 0;6 0 7 0 5 0 4;0 0 0 0 0 0 0;14 0 15 0 13 0 12;0 0 0 0 0 0 0;10 0 11 0 9 0 8]. Pengaturan parameter untuk blok *Direct Lookup Table* diperlihatkan pada Gambar 3.6.

Tabel 3.1. Tabel *Lookup* untuk Blok *Direct Lookup Table*

	Q	-3		-1		1		3
I	Input 2 / Input 1	0	1	2	3	4	5	6
-3	0	2	0	3	0	1	0	0
	1	0	0	0	0	0	0	0
-1	2	6	0	7	0	5	0	4
	3	0	0	0	0	0	0	0
1	4	14	0	15	0	13	0	12
	5	0	0	0	0	0	0	0
3	6	10	0	11	0	9	0	8



Gambar 3.6 Pengaturan parameter blok *Direct Lookup Table*

3.1.2.10 *Integer to Bit Converter*

Blok ini berfungsi untuk mengkonversi sinyal integer ke dalam bentuk biner. Jika parameter *Number of bits per integer* bernilai M, maka masukan integernya harus bernilai antara 0 dan $2^M - 1$. Blok ini memetakan setiap masukan integer ke dalam kelompok bit sejumlah M bit.

Pada rancangan, nilai integer yang menuju blok ini memiliki rentang antara 0 hingga 15 (2^4 - 1) sehingga nilai parameter *Number of bits per integer* adalah 4. Dengan demikian, maka untuk sebuah nilai integer akan dikonversi menjadi nilai biner sejumlah 4 bit. Misalkan nilai integernya [4,10], maka akan dikonversi menjadi [0,1,0,0 ; 1,0,1,0].

3.1.2.11 *Demux*

Blok ini berfungsi untuk mengekstrak komponen-komponen dari sinyal masukan menjadi sinyal-sinyal dengan komponen-komponen yang terpisah pada keluarannya. Sinyal keluarannya memiliki urutan dari *port* paling atas ke bawah.

Parameter *Number of outputs* menentukan jumlah *port* keluaran. Pada rancangan, parameter ini diberikan nilai 4 karena masukannya adalah nilai biner yang dikelompokkan tiap 4 bit.

3.1.2.12 *Counter*

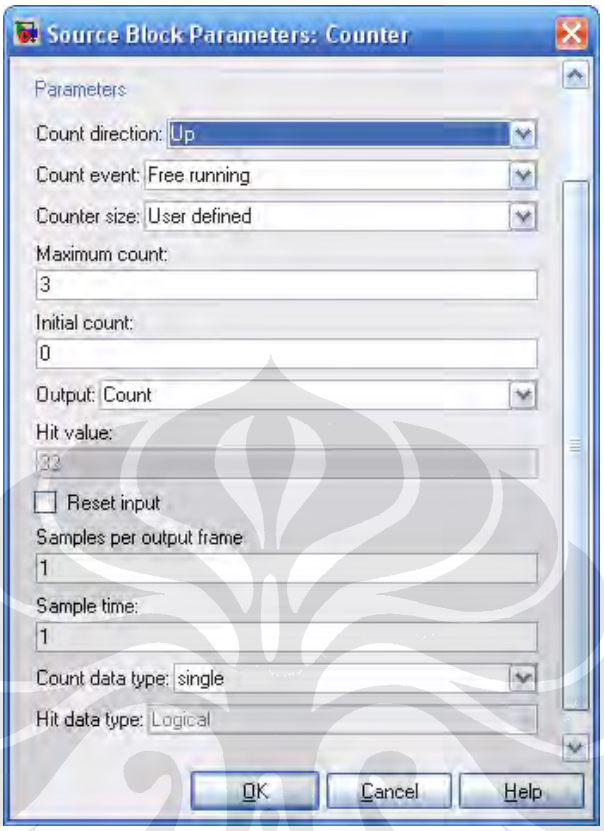
Blok ini berfungsi untuk melakukan perhitungan maju atau mundur melalui rentang nilai yang ditentukan. Parameter *Count direction* diatur *Up* agar *counter* menghirung maju. Parameter *Count event* diatur *Free running* agar *counter* menghitung secara kontinu dengan periode sesuai dengan yang ditentukan pada parameter *Sample time*.

Parameter *Counter size* menentukan rentang nilai integer yang harus dihitung sebelum kembali lagi ke nol. Parameter ini diatur *User defined* agar nilai maksimum bilangan integer yang dihitung dapat ditentukan melalui parameter *Maximum count*.

Parameter *Maximum count* diatur 3 dan *Initial count* diatur 0. Hal tersebut akan menyebabkan *counter* melakukan perhitungan secara terus menerus mulai dari 0 hingga 3 dan kemudian kembali lagi ke 0.

Parameter *Samples per output frame* menentukan jumlah sampel pada setiap frame dari keluaran. Jika keluarannya tidak menggunakan *frame-based output*, maka parameter ini diatur menjadi 1. Parameter *Sample time* menentukan periode sampel Ts dari keluaran (dalam mode *free-running*). Parameter *Sample*

time ini diatur 1 agar keluarannya melakukan erhitungan dengan periode 1. Pengaturan parameter untuk blok *Counter* ini diperlihatkan pada Gambar 3.7.



Gambar 3.7 Pengaturan parameter blok *Counter*

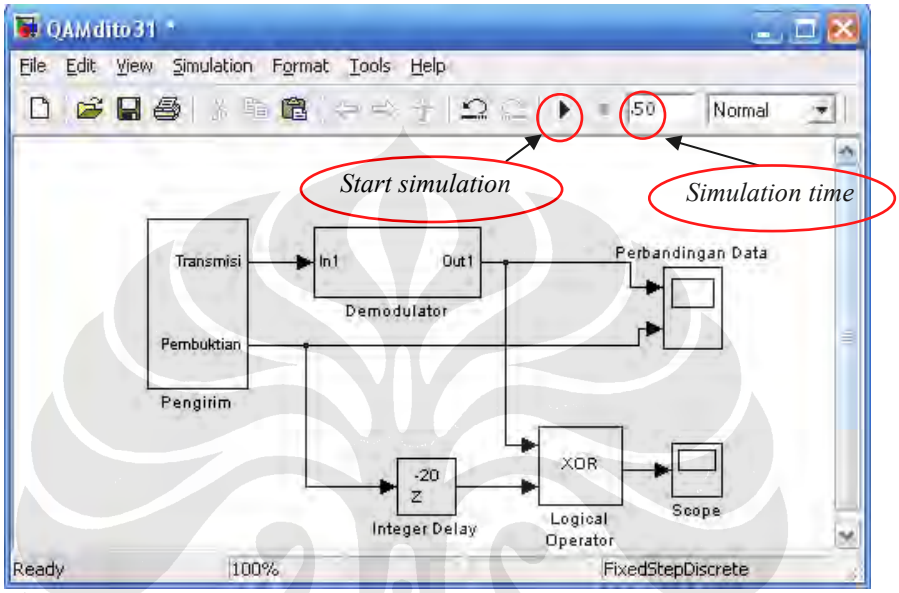
3.1.2.13 *Multiport Switch*

Blok ini berfungsi untuk memilih masukan mana yang harus diteruskan menuju keluaran. Masukan yang pertama disebut *control input* sedangkan masukan yang lainnya disebut *data input*. Nilai dari *control input* akan menentukan *data input* mana yang akan dilewatkan melalui *port* keluaran.

Parameter *Number of input* diatur 4 agar jumlah port *data input* berjumlah 4 buah. Data input dimasukkan setiap 4 bit, karena pada demodulasi 16-QAM nilai biner yang dihasilkan dikelompokkan setiap 4 bit. Selanjutnya dipilih *Use zero based indexing* karena *control input* menerima masukan dari blok *Counter* yang melakukan perhitungan dari 0 hingga 3.

3.1.3 Simulasi Sistem

Sebelum model hasil perancangan pada Simulink diimplementasikan pada papan DSK perlu dilakukan simulasi pada Simulink terlebih dahulu. Waktu simulasi diatur dengan memeberikan nilai *Simulation time* sebesar 50. Selanjutnya simulasi dapat dimulai dengan menekan tombol *start simulation* pada Simulink seperti ditunjukkan pada Gambar 3.8.



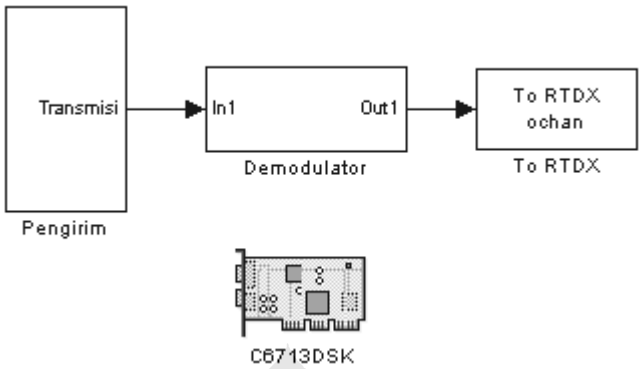
Gambar 3.8 Menjalankan simulasi sistem

Setelah simulasi selesai dijalankan, hasil simulasi tersebut dapat dilihat pada kedua buah *scope*. Jika *scope* "Perbandingan Data" di-*doubleclick* maka akan terlihat bentuk sinyal dari pengirim dan sinyal yang diterima. Selanjutnya jika *scope* setelah blok XOR di-*doubleclick* maka akan terlihat hasil operasi XOR antara sinyal yang dikirim dengan diterima.

3.2 RANCANG BANGUN SISTEM MENGGUNAKAN DSK TMS320C6713

Model yang digunakan untuk perancangan demodulator 16-QAM pada DSK TMS320C6713 hampir sama dengan model untuk perancangan dengan Simulink. Untuk perancangan ini, model juga dibuat dengan menggunakan Simulink hanya saja pada rancangan ini ditambahkan blok target *C6713DSK* dan

blok *To RTDX* serta tidak perlu menggunakan blok *scope.* Diagram blok dari sistem yang dibuat ditunjukkan pada Gambar 3.9.



Gambar 3.9 Diagram blok rancang bangun pada DSK TMS320C6713

Blok C6713DSK berfungsi menyediakan akses untuk pengaturan perangkat prosesor yang dibutuhkan untuk pembangkitan kode oleh *Real Time Workshop* untuk dijalankan pada target. Blok ini harus ditambahkan pada model Simulink jika ingin menggunakan target DSK C6713.

Keluaran dari demodulator adalah sinyal yang memiliki bentuk biner. Jika sinyal ini dikeluarkan melalui konektor *Line Out* pada papan DSK, maka akan terjadi konversi oleh DAC pada papan DSK yang mengakibatkan sinyal diubah ke dalam bentuk analog. Sinyal tersebut akan sulit diamati pada *oscilloscope* sehingga sulit untuk dianalisis. Sebagai alternatif dari permasalahan tersebut, maka dapat digunakan RTDX untuk mengamati sinyal keluaran dari demodulator.

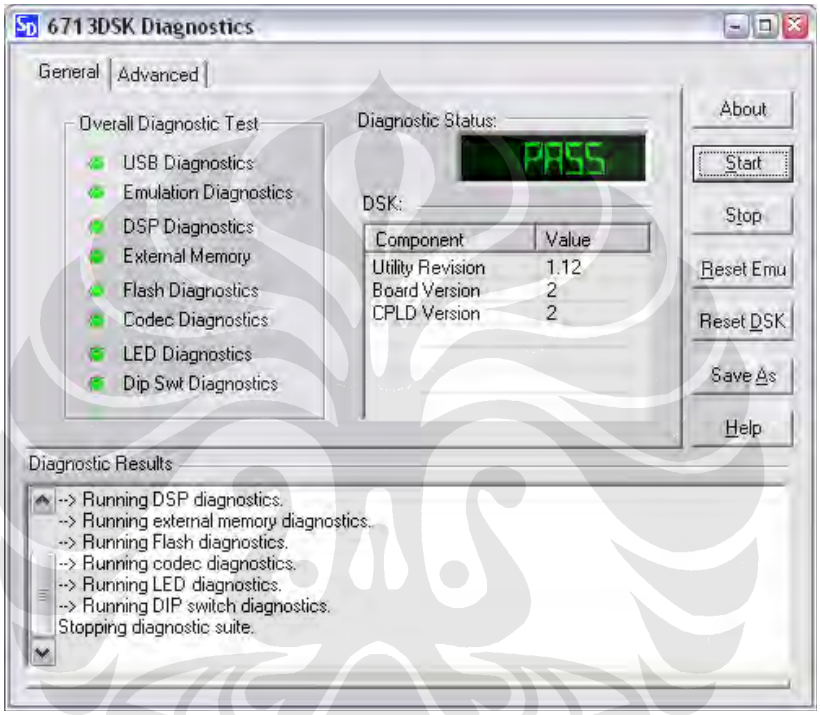
Untuk itu digunakan blok *To RTDX* yang berfungsi untuk membuat RTDX *output channel* pada target DSK. *Output channel* ini akan mentransfer data dari target menuju host secara *real time*. Pada simulasi, blok ini tidak melakukan operasi apapun. Blok ini akan menjalankan fungsinya untuk mentransfer data jika model dijalankan pada target DSK. Data tersebut dapat diperoleh dengan fungsi readmsg pada MATLAB.

Selain penambahan blok *C6713DSK* dan blok *To RTDX* pada Simulink, juga perlu dilakukan pengaturan-pengaturan tambahan yang akan dijelaskan pada bagian selanjutnya.

3.2.1 Konfigurasi Awal

Pemeriksaan awal yang perlu dilakukan yaitu:

1) Menjalankan *DSK Diagnostic Utility* yang tersedia pada CCS. Tes ini dilakukan untuk mengetahui apakah papan DSK terhubung dengan baik dan tidak menalami kerusakan. Jika semua tes berhasil dilewati, maka tampilannya akan seperti pada Gambar 3.10.



Gambar 3.10 Tampilan *CCS DSK Diagnostic utility*

2) Mengetikkan perintah **cssboardinfo** pada *command window* MATLAB untuk mengetahui apakah CCS terinstal pada komputer. Jika CCS terinstal, maka MATLAB akan menampilkan informasi seperti pada Gambar 3.11.

```
Board Board                      Proc Processor                  Processor
 Num  Name                       Num  Name                       Type
 ---  --------------------------  ---  --------------------------  --------------
 0    C6713 DSK                   0    CPU_1                      TMS320C6x1x
```

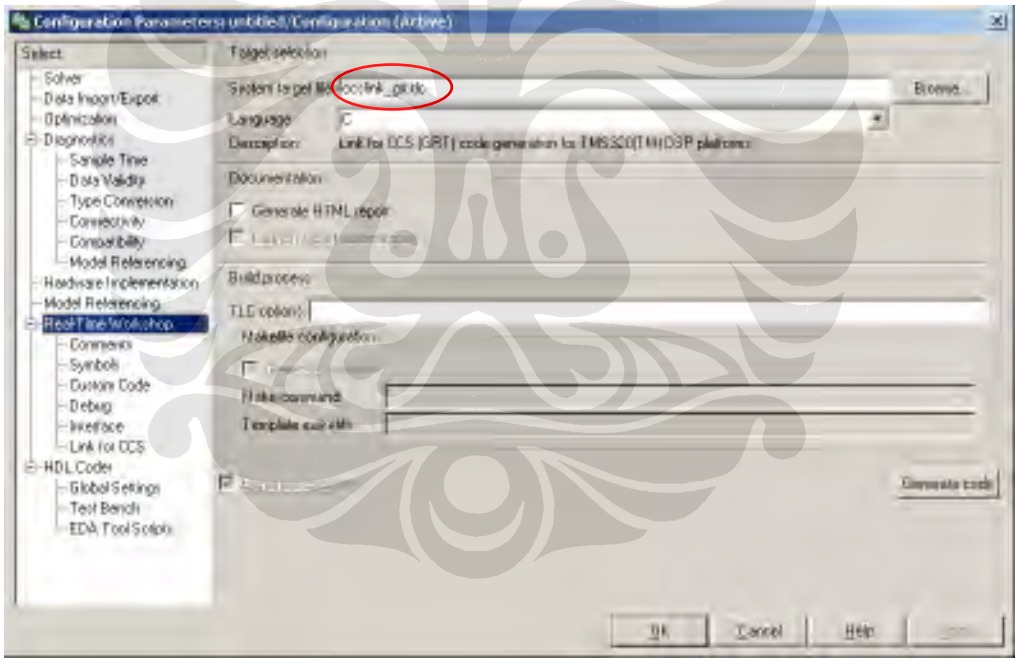
Gambar 3.11 Informasi yang ditampilkan pada MATLAB jika CCS telah terinstal

3) Mengetikkan perintah **c6000lib** pada *command window* MATLAB untuk memastikan terinstalnya *Embedded Target for TI C6000 DSP*. Jika terinstal, maka MATLAB akan menampilkan *library-library* untuk C6000.

3.2.2 Konfigurasi Parameter untuk Perangkat C6000

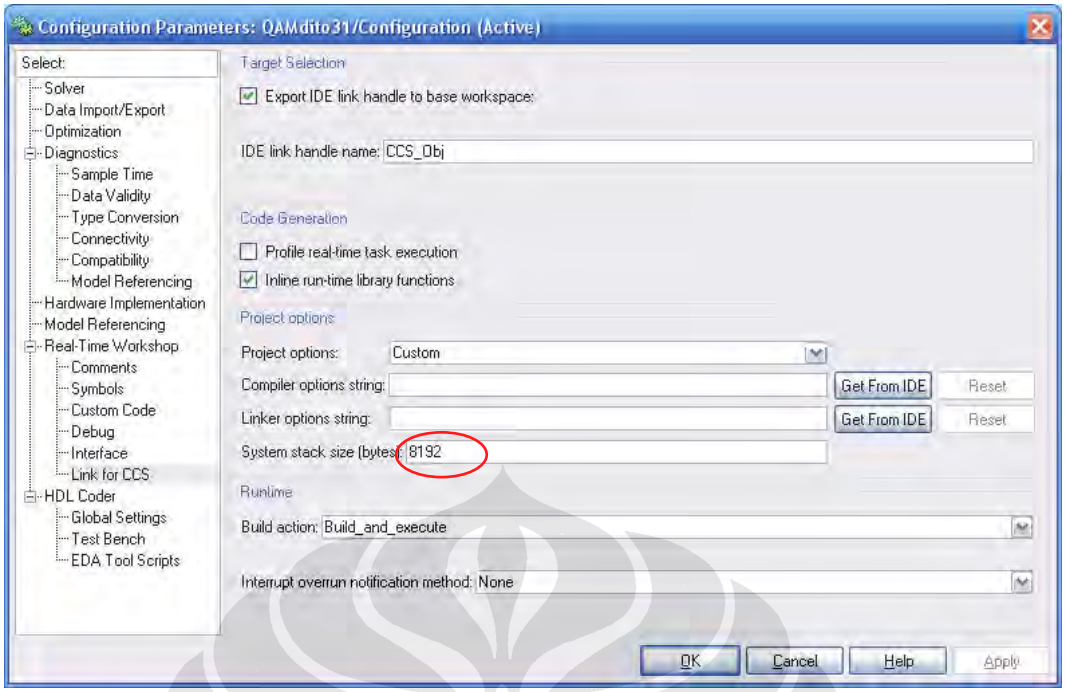
Berikut ini akan dijelaskan cara mengkonfigurasi Simulink dan *Real-Time Workshop* untuk digunakan dengan DSK C6713. Konfigurasi ini dapat dilakukan dengan cara memilih menu *Simulation* → *Configuration Parameters* pada Simulink.

Selanjutnya dipilih kategori *Real-Time Workshop* pada bagian sebelah kiri. Pada bagian *Target selection* (sebelah kanan), *system target file* diubah menjadi **ccslink_grt.tlc** seperti ditunjukkan pada Gambar 3.12.



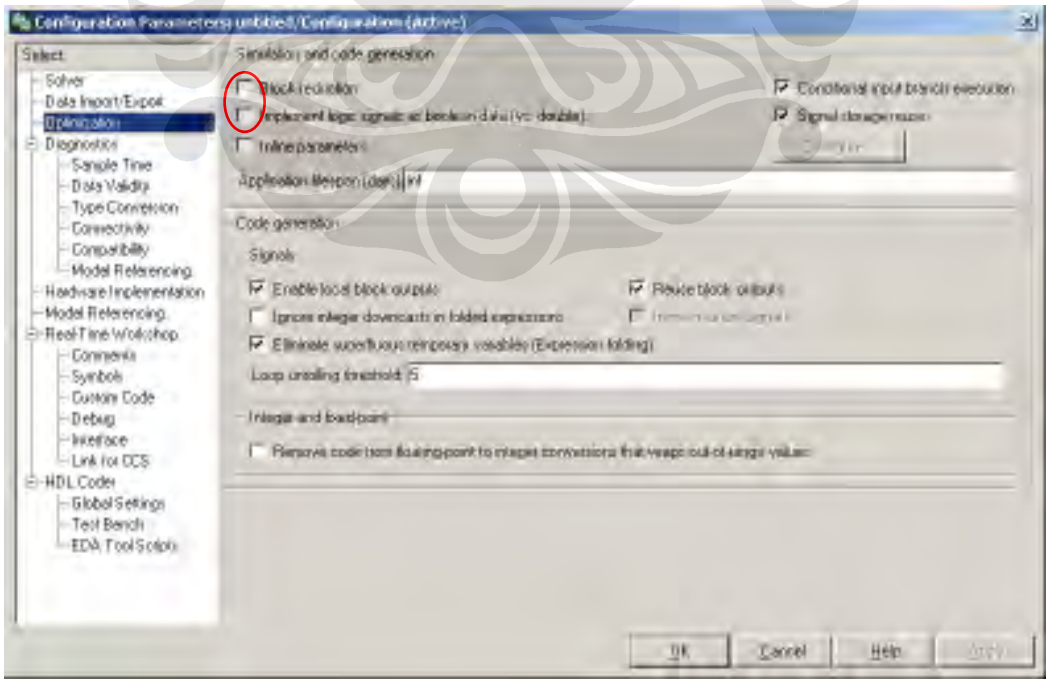
Gambar 3.12 Mengubah *system target file*

Pada kategori *Real-Time Workshop,* dipilih bagian *Link for CCS*. Pada sebelah kanan, *System stack size* diubah menjadi 8192. Ukuran ini adalah ukuran yang sesuai dengan papan DSP. Cara mengubah *system stack size* tersebut ditunjukkan pada Gambar 3.13.



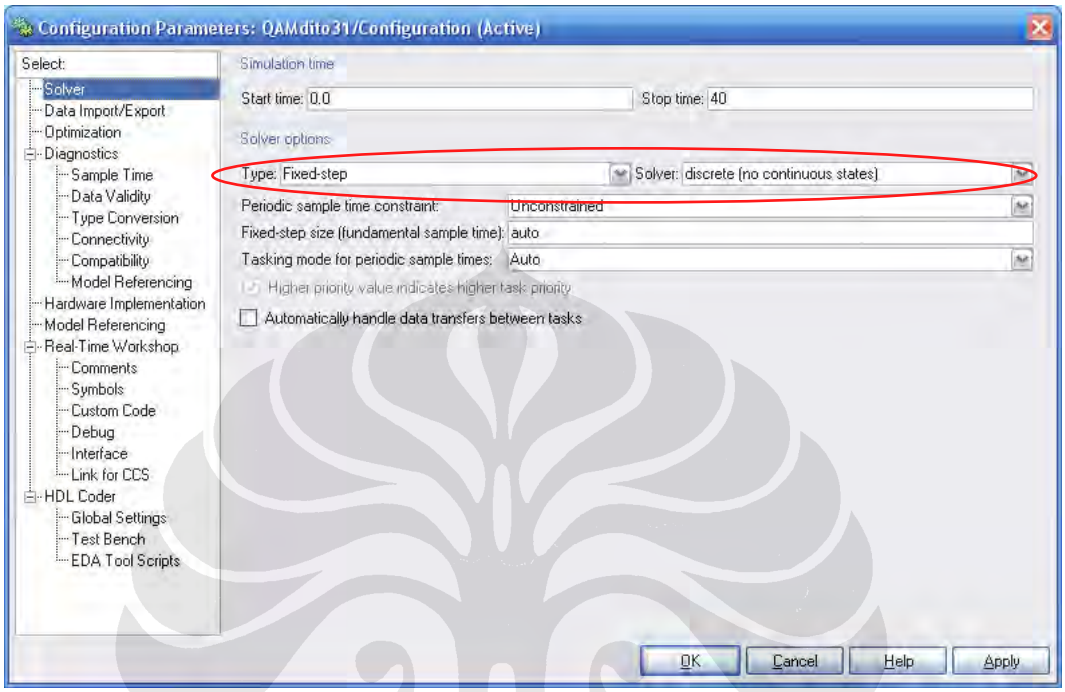
Gambar 3.13 Mengubah *system stack size*

Pada kategori *Optimization,* untuk *Simulation and code generation* pilihan pada *Block reduction optimization* dan *Implement logic signals as Boolean data* dihilangkan tandanya. Pengaturan ini ditunjukkan pada Gambar 3.14.



Gambar 3.14 Mengubah pengaturan pada bagian *optimization*

Pada kategori *Solver*, pilihan *Solver option* pada bagian *Type* diubah menjadi *Fixed-step* dan bagian *Solver* diubah menjadi *discrete*. Pengaturan ini ditunjukkan pada Gambar 3.15.

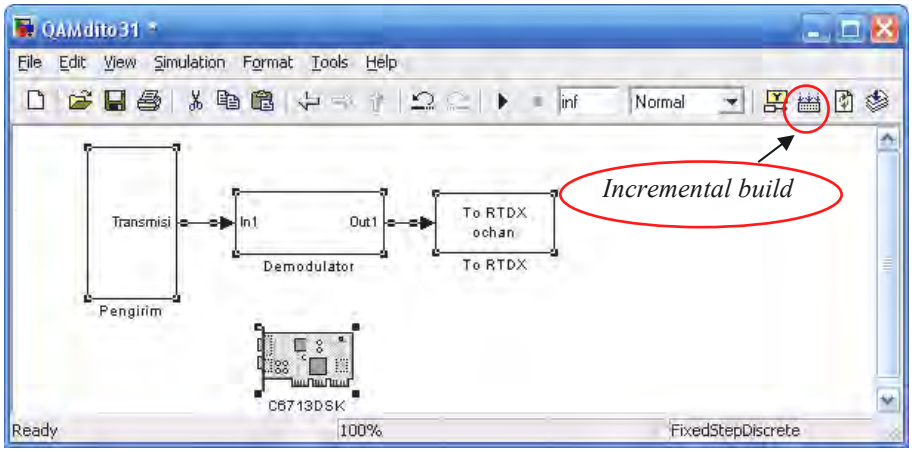


Gambar 3.15 Mengubah *solver* menjadi *discrete*

3.2.3 Implementasi Sistem pada DSK C6713

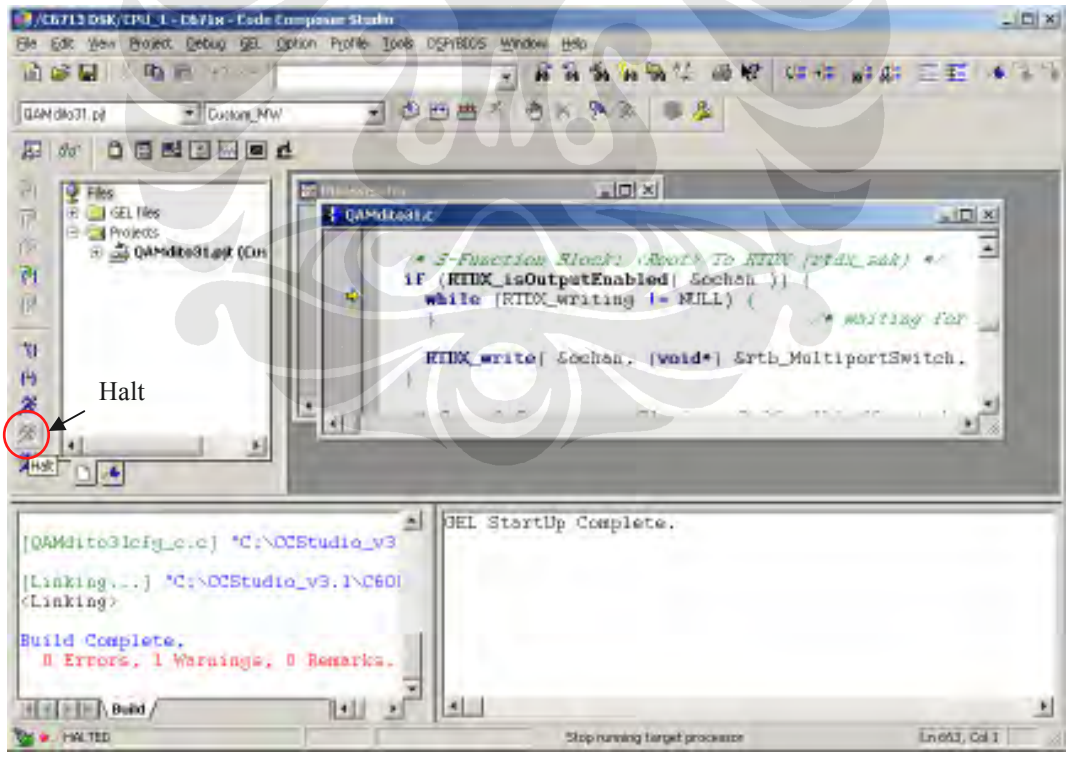
3.2.3.1 *Mendownload dan menjalankan sistem pada DSK C6713*

Setelah model pada Simulink selesai dibuat dan konfigurasi telah selesai dilakukan, maka model tersebut kemudian dapat didownload ke dalam target DSK C6713. Untuk mengimplementasikan model Simulink ke dalam target DSK C6713 dapat dilakukan dengan mengklik *toolbar button Incremental build* pada model Simulink yang dibuat seperti ditunjukkan pada Gambar 3.16. Dengan mengklik *icon* tersebut, maka Simulink akan membangun CCS *project* dan selanjutnya CCS akan mengkompilasinya sedemikian rupa sehingga program dapat dijalankan langsung dari DSK.



Gambar 3.16 Membangun model ke dalam DSK

Demodulator yang telah berjalan pada DSK tersebut dapat dikendalikan dengan bantuan CCS. Untuk menghentikan sementara proses pada DSK dapat dilakukan dengan menekan tombol *Halt* pada CCS seperti ditunjukkan pada Gambar 3.17.

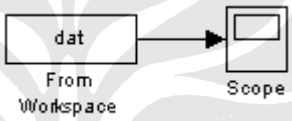


Gambar 3.17 Menghentikan sementara proses yang berlangsung pada DSK

3.2.3.2 *Memperoleh data dari DSK dengan menggunakan RTDX*

Untuk memperoleh data keluaran dari demodulator yang dijalankan pada DSK digunakan bantuan *file* bernama RTDXdriver.m. Untuk menggunakan *file* tersebut dilakukan dengan cara mengetik RTDXdriver('*path/filename*') pada *command window*. Pada file tersebut terdapat fungsi RTDXdriver yang akan melakukan serangkaian prosedur untuk membaca data dari *RTDX output channel*. Data tersebut kemudian akan disimpan ke dalam sebuah *array* di dalam file berekstensi .mat.

Untuk membaca dan memplot data yang telah diperoleh tersebut ke dalam grafik dilakukan dengan cara membuat model baru pada Simulink seperti pada Gambar 3.18.



Gambar 3.18 Model untuk memplot data dari *array* dat

Parameter *Data* pada blok *From workspace* diatur menjadi dat dan *sample time* disesuaikan dengan data keluaran demodulator yaitu sebesar 1. Sebelum model tersebut dijalankan, pastikan bahwa file berekstensi .mat yang dihasilkan oleh RTDXdriver tadi telah dibuka dan pada *workspace* telah muncul *array* bernama dat. Selanjutnya dengan menjalankan model tersebut, maka data dari *array* tersebut akan ditampilkan pada *scope*.

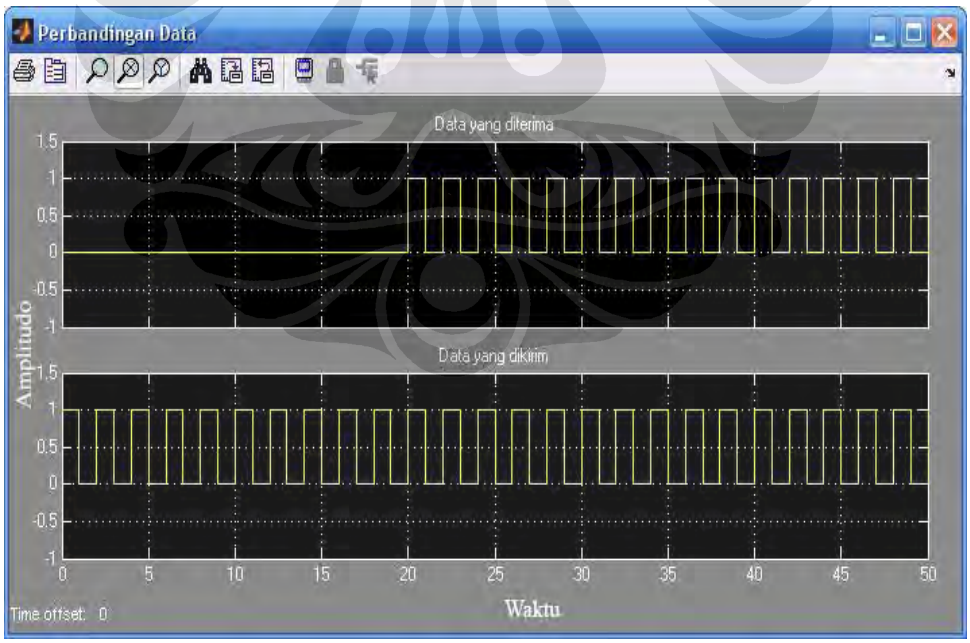
BAB IV
UJI COBA DAN ANALISIS

4.1 UJI COBA RANCANG BANGUN DENGAN SIMULINK

Uji coba hasil rancang bangun sistem pada Simulink dilakukan dengan menjalankan simulasi model rancangan sistem demodulator 16-QAM pada Simulink. Setelah simulasi dijalankan, maka akan diperoleh data berupa bentuk sinyal keluaran hasil demodulasi yang dapat dilihat pada *scope*. Bentuk sinyal keluaran tersebut diamati selama waktu simulasi 50 detik.

4.1.1 Sinyal Informasi Berbentuk Pulsa-Pulsa Gelombang Persegi

Pada simulasi yang pertama digunakan sinyal informasi berupa sinyal berbentuk pulsa-pulsa gelombang persegi. Dari hasil simulasi tersebut, bentuk sinyal keluaran demodulator dan sinyal yang dikirimkan diperoleh seperti pada Gambar 4.1.

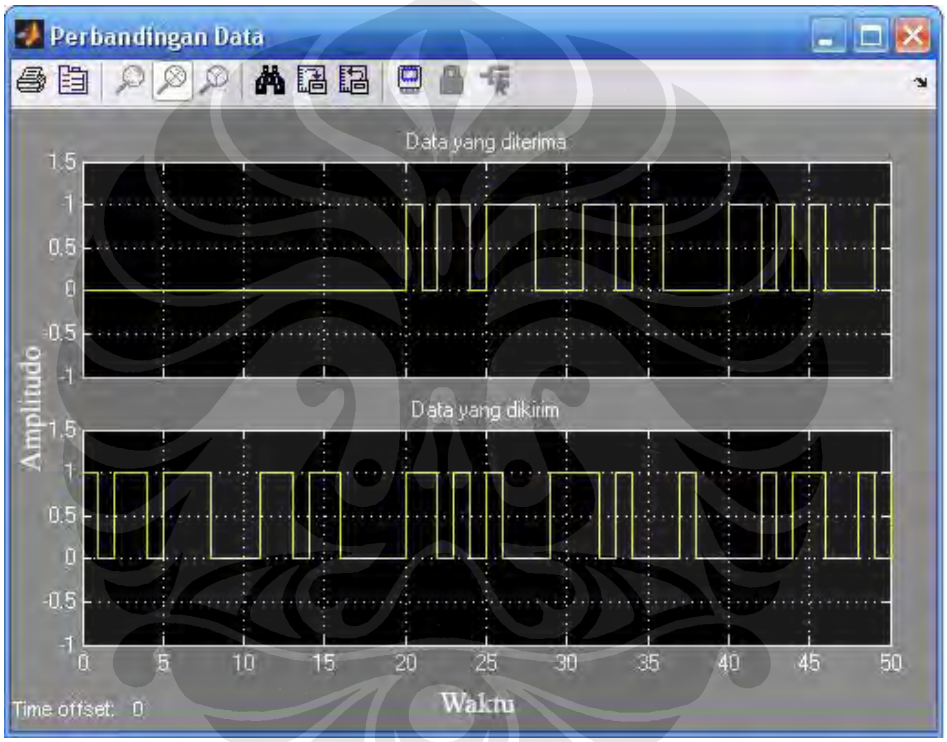


Gambar 4.1 Bentuk sinyal hasil demodulasi dan sinyal yang dikirim untuk sinyal informasi berbentuk pulsa gelombang persegi

Pada Gambar 4.1 dapat dilihat bahwa sinyal dari sisi pengirim dan sinyal hasil demodulasi memiliki bentuk yang sama persis, hanya saja sinyal hasil demodulasinya mengalami *delay* selama 20 detik. *Delay* tersebut dapat timbul akibat adanya *delay* pada sisi pengirim dan juga *delay* pada sisi penerima.

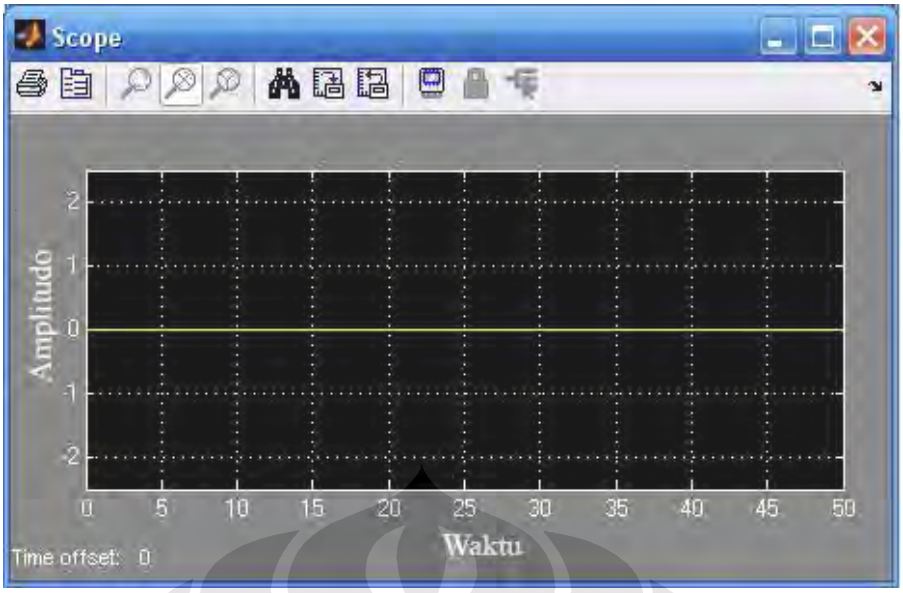
4.1.2 Sinyal Informasi Berupa Sinyal Biner Acak

Pada simulasi yang kedua digunakan sinyal informasi sumber berupa sinyal biner acak. Pada simulasi ini diperoleh data seperti yang ditunjukkan pada Gambar 4.2.



Gambar 4.2 Bentuk sinyal hasil demodulasi dan sinyal yang dikirim untuk sinyal informasi berupa sinyal biner acak

Jika Gambar 4.2 dilihat secara sepintas, maka agak sulit untuk membandingkan bentuk dari kedua sinyal tersebut. Oleh karena itu, untuk memudahkan dalam melakukan perbandingan sinyal yang dikirim dengan sinyal hasil demodulasi, maka digunakan blok *Logical operator XOR*. Hasil keluaran dari blok XOR yang ditunjukkan pada *scope* untuk simulasi ini ditunjukkan pada Gambar 4.3.



Gambar 4.3 Bentuk sinyal keluaran dari blok XOR

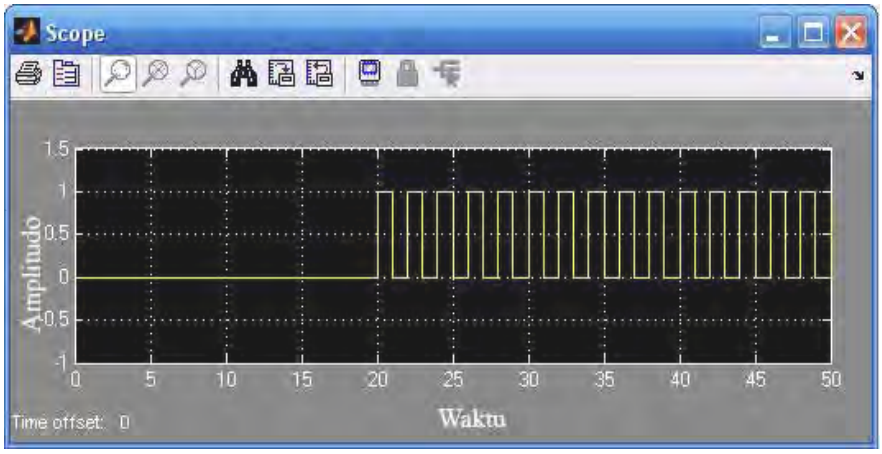
Pada Gambar 4.3 terlihat bahwa sinyal keluaran dari blok XOR tersebut merupakan sinyal yang selalu bernilai nol. Hal tersebut menunjukkan bahwa sinyal yang dikirim dengan sinyal hasil demodulasi memiliki bentuk yang sama, karena blok XOR hanya menghasilkan nilai nol jika kedua buah masukkannya memiliki nilai yang sama persis.

4.2 UJI COBA RANCANG BANGUN DENGAN DSK TMS320C6713

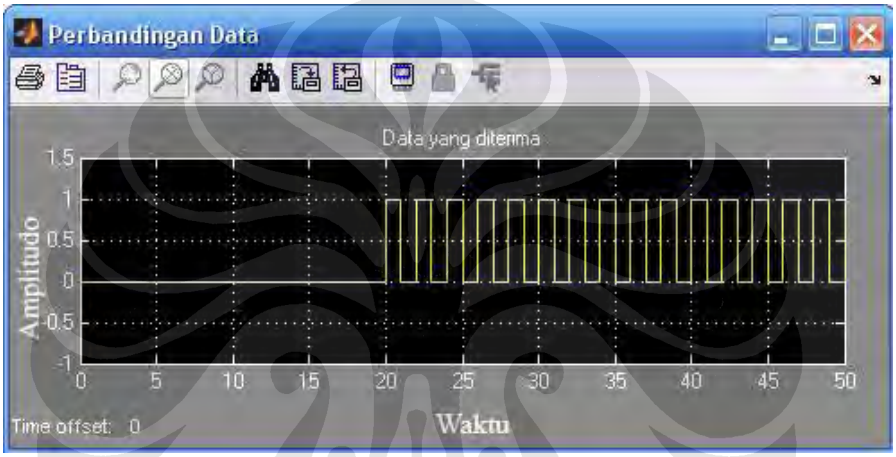
Uji coba hasil rancang bangun demodulator pada DSK TMS320C6713 dilakukan dengan terlebih dahulu menjalankan model demodulator pada papan DSK. Pengambilan data pada uji coba ini dilakukan dengan bantuan *file* "RTDXdriver.m". Data yang diperoleh berada di dalam *file* berekstensi .mat yaitu berupa *array* dengan nama dat.

4.2.1 Sinyal Informasi Berbentuk Pulsa-Pulsa Gelombang Persegi

Untuk sinyal informasi berupa sinyal berbentuk pulsa-pulsa gelombang persegi, maka diperoleh data sinyal keluaran pada *array* dat yang jika diplot pada *scope* akan tampak seperti pada Gambar 4.4(a). Jika hasil tersebut dibandingkan dengan hasil simulasi maka terlihat bahwa hasil uji coba pada DSK menunjukkan hasil yang sama dengan simulasi seperti ditunjukkan pada Gambar 4.4.



(a)

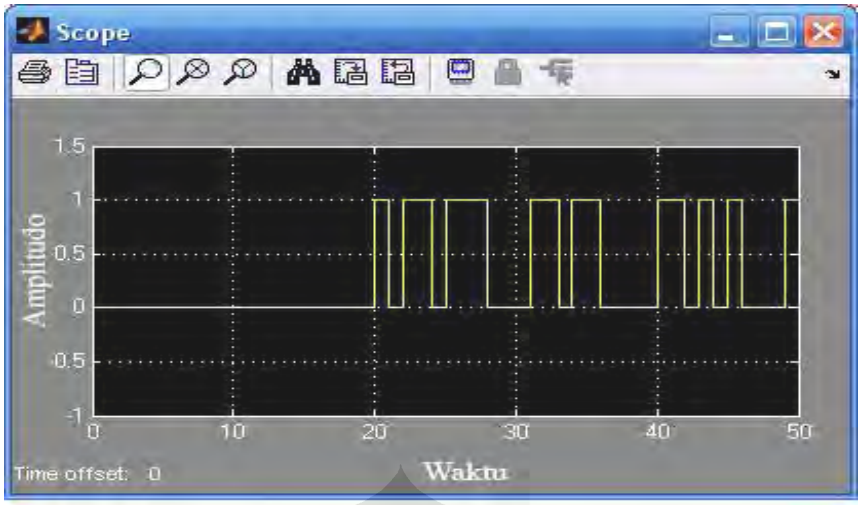


(b)

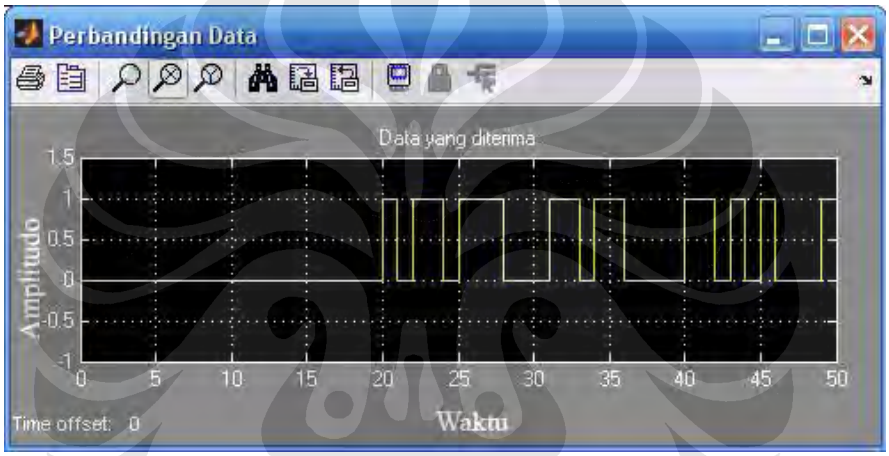
Gambar 4.4 Perbandingan hasil uji coba pada DSK TMS320C6713 dengan hasil simulasi untuk sinyal informasi berbentuk pulsa-pulsa gelombang persegi

4.2.2 Sinyal Informasi Berupa Sinyal Biner Acak

Hasil uji coba pada DSK untuk sinyal informasi berupa sinyal biner acak ditunjukkan pada Gambar 4.5(a). Perbandingan antara hasil uji coba pada DSK dengan hasil simulasi ditunjukkan pada Gambar 4.5. Pada Gambar 4.5 terlihat bahwa hasil yang ditunjukkan oleh keduanya memperlihatkan hasil yang sama. Dengan demikian dapat dikatakan bahwa demodulator yang dibangun pada DSK TMS320C6713 telah berfungsi secara ideal seperti yang telah disimulasikan pada Simulink.



(a)



(b)

Gambar 4.5 Perbandingan hasil uji coba pada DSK TMS320C6713 dengan hasil simulasi untuk sinyal informasi berupa sinyal biner acak

BAB V
KESIMPULAN

Dari pembahasan tentang demodulator 16-QAM yang telah dibangun pada DSK TMS320C6713 dan pengujiannya, dapat disimpulkan bahwa demodulator 16-QAM dapat dibangun pada DSK TMS320C6713 namun tidak dapat menggunakan masukan melalui *port line-in*.

DAFTAR ACUAN

[1] Hewlett-Packard Company, "Digital Modulation in Communications Systems – An Introduction", U.S.A, Juli 1997.

[2] National Instruments, "Quadrature Amplitude Modulation (QAM)", http://zone.ni.com/devzone/cda/tut/p/id/3896.htm, 30.11.2007

[3] Endra, Fauzie, "Penggunaan Prosesor Sinyal Digital Tms320C542 Untuk Simulasi Pengiriman Data Melalui Frekuensi *Voice Band* Menggunakan *Quadrature Amplitude Modulation*", *Proceedings, Komputer dan Sistem Intelijen (KOMMIT2004),* pp.727-736, Universitas Gunadarma, Jakarta, 24 – 25 Agustus 2004.

[4] Wikipedia Foundation, Inc., "Quadrature Amplitude Modulation", http://www.wikipedia.org, 30.08.2007.

[5] Washington, G., Rajagopalan, A., "Simulink® Tutorial", The Intelligent Structures and System Laboratory Department of Mechanical Engineering The Ohio-State University, 2003.

[6] Ycchan, "Modern Communication Systems: Tutorial 1 – Introduction to Simulink", 2005.

[7] Chassaing, Rulph, "Digital Signal Processing and Applications with the C6713 and C6416 DSK*",* John Willey & Sons.inc, 2005.

[8] Brown, D. R., "Digital Signal Processing and Applications with the TMS320C6713 DSK", October 15-16, 2007

[9] Hasnain, S. K., Jamil, N., "Implementation of DSP Real Time Concepts Using Code Composer Studio 3.1, TI DSK TMS320C6713 and DSP Simulink Blocksets".

DAFTAR PUSTAKA

Intel Corporation, “Adaptive Modulation (QPSK, QAM)”, USA, 2004.

Spectrum Digital, Inc., “TMS320C6713 DSK Technical Reference”, 2003.

Langton, C., “All About Modulation”, www.complextoreal.com, 2002.

MathWorks, Inc., “Communications Blockset 2”, 2000.

Altera, “QAM: Quadrature Amplitude Modulation”, http://www.altera.com/products/ip/t-alt-qam.html, 25.09.2007.

LAMPIRAN

LAMPIRAN 1
Listing file "RTDXdriver.m"

```
function RTDXdriver(modelname)
% RTDXDRIVER Reads and plots data through an RTDX channel.

[modelpath,modelname,modelext] = fileparts(modelname);

cc = ccsdsp;
set(cc,'timeout',50);
if ~isrtdxcapable(cc)
    error('Processor does not RTDX support');
end

cc.reset; pause(1);
cc.cd(modelpath);
cc.visible(1);

open(cc,sprintf('%s.pjt',modelname));
load(cc,sprintf('%s.out',modelname));

rx = cc.rtdx;
rx.set('timeout', 50); % Reset timeout = 10 seconds

rx.configure(64000,4);

rx.open('ochan','r');

rx.enable; % enable RTDX

cc.run; % cc.enable can be placed here

pause(1); % cc.enable cannot be placed here; too much time had
passed
          % RTDX processing will be 'stalled'

% source array preparing
ukuran=601;
waktu_cuplik=1;
i=1;
enable(rx,'ochan');
while (i<ukuran)
       if isenabled(rx,'ochan')
           dat(i,2)=readmsg(cc.rtdx,'ochan','double');
       end
    i=i+1;
end
i=1;
for a=0:waktu_cuplik:((ukuran-2)*waktu_cuplik)
dat(i,1)=a;
i=i+1;
end
RTDXcleanup(cc,rx);
            matfile=strcat(modelname,'.mat');
            save(matfile,'dat');
```

```
%=========================================================================
=========
% Put RTDX back to good state
%=========================================================================
=========
function RTDXcleanup(cc,rx)
if isrunning(cc),          % if the target DSP is running
    halt(cc);              % halt the processor
end

cc.reset;

disable(rx,'ochan');
disable(rx);                % disable RTDX

close(cc.rtdx,'ochan');
```