RANCANG BANGUN MODULATOR 16-QAM PADA DSK TMS 320C6713 DENGAN MENGGUNAKAN SIMULINK

SKRIPSI

Oleh

RUNDU ADI WAHYUDI

04 04 03 075 X

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GENAP 2007/2008

RANCANG BANGUN MODULATOR 16-QAM PADA DSK TMS 320C6713 DENGAN MENGGUNAKAN SIMULINK

SKRIPSI

Oleh

RUNDU ADI WAHYUDI

04 04 03 075 X

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 MODULATOR 16-QAM
PADA DSK TMS 320C6713
DENGAN MENGGUNAKAN 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

Rundu Adi Wahyudi
NPM 04 04 03 075 X

PENGESAHAN

Skripsi dengan judul :

RANCANG BANGUN MODULATOR 16-QAM
PADA DSK TMS 320C6713
DENGAN MENGGUNAKAN 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 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

Penulis 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.

Rundu Adi Wahyudi
NPM 04 04 03 075 X
Departemen Teknik Elektro

Dosen Pembimbing
Dr. Ir. Arman Djohan, M.Eng

RANCANG BANGUN MODULATOR 16-QAM PADA DSK TMS C6713 DENGAN MENGGUNAKAN SIMULINK

ABSTRAK

Tiga hal yang paling mendasar pada sistem komunikasi yaitu efisiensi *bandwidth*, efisiensi daya, dan efisiensi biaya. Efisiensi *bandwidth* menjadi prioritas yang paling utama pada kebanyakan sistem komunikasi. Oleh karena itu, modulasi digital digunakan sebagai ganti dari modulasi analog untuk mengatasi masalah keterbatasan *bandwidth*. Salah satu pola modulasi digital yang banyak digunakan adalah *Quadrature Amplitude Modulation* (QAM).

QAM adalah suatu teknik modulasi yang mengirimkan data dengan cara mengubah amplitudo dari dua gelombang carrier yang mempunyai beda fase sebesar 90°. Dalam bentuk *hardware*, modulator QAM yang ada saat ini biasanya dalam bentuk *chip* atau mikroprosesor. Cara lain untuk membuat modulator dalam bentuk *software* yang dapat diprogram yaitu dengan menggunakan DSP *processor*. Untuk mengimplementasikannya, pada skripsi ini dilakukan rancang bangun 16-QAM pada DSK TMS320C6713 dengan menggunakan Simulink.

Pemrograman DSP *processor* secara manual yang rumit dapat dihindari dengan menggunakan Simulink. Model sistem 16-QAM modulator yang akan dibuat dirancang pada Simulink, kemudian disimulasikan. Setelah itu, dengan menggunakan pustaka C6713DSK yang tersedia pada Simulink, model sistem yang telah dibuat dibangkitkan kode programnya secara otomatis dan dimuat ke dalam C6713DSK melalui perantara *software* Code Composer Studio.

Analisis dilakukan dengan membandingkan hasil keluaran rancang bangun pada Simulink dan DSK TMS320C6713 dengan menggunakan osiloskop. Dari hasil perbandingan, terdapat sedikit perbedaan bentuk gelombang akibat adanya *noise*. *Noise* ini disebabkan oleh *grounding* yang kurang baik pada *probe* osiloskop. Namun, secara umum bentuk kedua gelombang yang dibandingkan sama sehingga dapat disimpulkan bahwa rancang bangun ini telah berfungsi sebagaimana mestinya.

Kata kunci : 16-QAM, Simulink, DSK TMS320C6713, DSP

Rundu Adi Wahyudi
NPM 04 04 03 075 X
Electrical Engineering Department

Counsellor
Dr. Ir. Arman Djohan, M.Eng

16-QAM MODULATOR DESIGN ON DSK TMS320C6713 USING SIMULINK

ABSTRACT

Three fundamentals problem in the communication system was bandwidth efficiency, power efficiency, and cost efficiency. In most systems, a high priority was on bandwidth efficiency. To solved this problem, digital modulation was use to replace analog modulation. One of digital modulation scheme that generally used was Quadrature Amplitude Modulation (QAM).

QAM is a modulation scheme which conveys data by changing (*modulating*) the amplitude of two carrier waves which out of phase with each other by 90°. In hardware form, QAM modulator usually made as a chip or microprocessor. Another way to built this device was using a DSP processor. To implement the DSP processor, in this thesis, 16-QAM modulator was designed on DSK TMS320C6713 by means of Simulink software.

Simulink was used to avoid the use of algorithm such as assembly and C. The model of 16-QAM modulator was designed and simulated on Simulink. Then by using C6713DSK library, program code of the model was generated and downloaded to the C6713DSK pass through the Code Composer Studio.

The analysis was done by compared the design result of Simulink and DSK TMS320C6713 using oscilloscope. The result, a little differences was seen in the wave form caused by noise. This noise was occured because of bad grounding at the oscilloscope probe.Generally, the form of that two wave was exactly the same. So the conclusion, the 16-QAM modulator was successfully implemented on DSK TMS320C6713.

Keywords : 16-QAM, Simulink, DSK TMS320C6713, DSP

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

ADC	Analog to Digital Converter
ALU	Arithmatic Logic Unit
AM	Amplitude Modulation
API	Application Programming Interface
CCS	Code Composer Studio
DAC	Digital to Analog Converter
DSK	DSP Starter Kit
DSP	Digital Signal Processing
FIR	Finite Impulse Response
IDE	Integrated Development Environment
QAM	Quadrature Amplitude Modulation
RTDX	Real Time Data Exchange
TI	Texas Instrument
USB	Universal Synchronous Bus
VLIW	Very Long Instruction Word

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Sistem komunikasi akan selalu berkaitan dengan tiga hal yaitu efisiensi *bandwidth*, efisiensi daya, dan efisiensi biaya. Efisiensi *bandwidth* menjadi prioritas utama pada sistem komunikasi saat ini. Hal ini berkenaan dengan kemampuan dari pola modulasi untuk mengakomodasi data pada *bandwidth* yang terbatas. Modulasi digital digunakan untuk menggantikan modulasi analog karena menawarkan kapasitas yang lebih besar dibandingkan dengan modulasi analog dalam pengiriman informasi pada *bandwidth* yang terbatas [1].

Salah satu teknik dari modulasi digital adalah *Quadrature Amplitude Modulation* (QAM) yang merupakan teknik modulasi digital multisimbol. Modulasi digital multisimbol mempunyai keunggulan yaitu kecepatan yang lebih tinggi karena setiap simbol yang dikirimkan melambangkan beberapa bit sekaligus. Aplikasi QAM banyak digunakan pada *microwave radio digital*, televisi kabel digital, modem, dan lain-lain.

Modulator atau demodulator QAM dalam bentuk hardware biasanya berupa *chip* atau *microprocessor*. Tentunya sangat sulit untuk membuat chip ini karena memerlukan proses fabrikasi yang kompleks. Cara lain yang lebih mudah untuk membuat modulator atau demodulator QAM adalah dengan melakukan pemrograman *Digital Signal Processing* (DSP) *processor*.

DSP *processor* digunakan dalam berbagai macam aplikasi, mulai dari komunikasi dan kontrol hingga *speech* dan *image processing*. Pemrograman DSP *processor* dilakukan menggunakan bahasa C dan bahasa *assembly*, melalui software *Code Composer Studio* (CSS). Pemrograman DSP *processor* secara manual ini sangat rumit dan menyita waktu.

Metode pemrograman lain yang lebih praktis yaitu menggunakan DSP Simulink *blockset* yang terdapat dalam *software* MATLAB. Dengan menggunakan *Link to* C6000 DSP yang tersedia pada *library* Simulink, model

sistem yang telah dibuat dapat dibangkitkan kode programnya secara otomatis dan dimuat ke dalam C6000 DSP *processor*.

1.2 TUJUAN PENULISAN

Tujuan dari skripsi ini adalah melakukan rancang bangun modulator 16-QAM pada DSP *processor* TMS320C6713 dengan menggunakan SIMULINK.

1.3 BATASAN MASALAH

Masalah yang akan dibahas pada skripsi ini dibatasi pada rancang bangun modulator 16-QAM pada DSK TMS320C6713 menggunakan model Simulink tanpa pemrograman bahasa C secara manual. Pada uji coba, sumber sinyal tidak menggunakan sinyal generator dari perangkat laboratorium. Dalam analisis, tidak dilakukan penghitungan besarnya *noise* yang terjadi pada output sistem.

1.4 METODOLOGI PENELITIAN

Skripsi ini disusun berdasarkan studi literatur dari berbagai macam sumber yang relevan terhadap penelitian, pemodelan sistem berdasarkan parameter-parameter yang diketahui dan analisis terhadap konsep-konsep mengenai implementasi terhadap perancangan model pada DSK ini.

1.5 SISTEMATIKA PENULISAN

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

1. BAB 1 PENDAHULUAN

   Bab ini menjelaskan latar belakang, tujuan penulisan, pembatasan masalah, metodologi penulisan dan sistematika penulisan.

2. BAB 2 TINJAUAN PUSTAKA

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

3. BAB 3 RANCANG BANGUN SISTEM 16-QAM

   Bab ini menjelaskan pemodelan sistem 16-QAM pada Simulink, *setting* parameter model, dan perancangan sistem pada DSK TMS320C6713.

4. BAB 4 UJI COBA DAN ANALISIS

   Bab ini menjelaskan uji coba yang dilakukan dan menganalisis hasil output dari Simulink dan DSK TMS320C6713.

5. BAB 5 KESIMPULAN

   Bab ini berisi kesimpulan atas perancangan yang dilakukan.

BAB II
QAM, SIMULINK, DSK TMS320C6713, DAN CCS

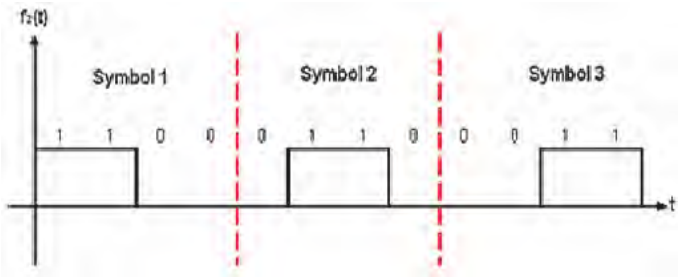
2.1 *QUADRATURE AMPLITUDE MODULATION* (QAM)

QAM adalah kombinasi dari *amplitude modulation* dan *phase modulation* dengan bit rate sebesar 2400 bit/detik atau lebih. Aliran data yang akan ditransmisikan dibagi menjadi sekumpulan bit, bergantung dari orde QAM, kemudian ditumpangkan ke dua *carrier* yang mempunyai frekuensi yang sama namun mempunyai beda fase antara keduanya sebesar 90^0 [2]. Kata *quadrature* pada QAM berasal dari kedua *carrier* yang berbeda fase 90^0.

$$s(t) = I(t)\cos(2\pi f_0 t) + Q(t)\sin(2\pi f_0 t) \quad \text{.....(2.1)}$$

Dari persamaan 2.1, dapat dilihat bahwa sinyal QAM dapat dibentuk dengan menjumlahkan sebuah sinyal kosinus dengan amplitudo I(t) dan sebuah sinyal sinus dengan amplitudo Q(t). Hal ini sama dengan menjumlahkan sebuah sinyal *Amplitude Modulation* (AM) yang menggunakan *carrier* kosinus dengan sebuah sinyal AM lain yang menggunakan *carrier* sinus. I(t) dan Q(t) adalah sinyal yang akan dimodulasi dan f_0 adalah frekuensi *carrier*.

Data yang akan dikirim dibagi menurut jumlah bit kemudian dikonversi menjadi suatu simbol seperti pada Gambar 2.1. Sebagai contoh untuk 16QAM (2^4 bit) maka akan terdapat 4 bit untuk satu simbol. Setelah itu, data yang telah dibagi dipetakan menurut diagram konstelasi dengan menggunakan *mapper*. Keluaran dari *mapper* adalah komponen *inphase* dan *quadrature* untuk simbol yang ditentukan oleh data tadi.



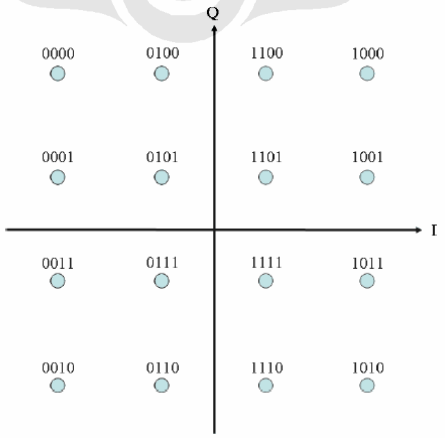
Gambar 2.1 Simbol 16-QAM yang terdiri dari 4 bit [3]

Dengan menggunakan orde yang lebih tinggi maka memungkinkan untuk mentransmisikan lebih banyak bit per simbol. Namun, jika rata-rata energi dari konstelasi tetap bernilai sama, maka jarak titik akan semakin berdekatan dan akan menyebabkan mudah terjadi *noise* dan perubahan lainnya sehingga menghasilkan *bit error rate* yang lebih tinggi.

2.1.1 Diagram Konstelasi

Diagram konstelasi adalah sebuah diagram dua dimensi yang merepresentasikan pola modulasi digital pada bidang kompleks. Tampilan diagram konstelasi dapat dilihat pada Gambar 2.2. Titik-titik pada diagram tersebut diurutkan berdasarkan aturan kode Gray.

Kode Gray adalah pengurutan nilai biner dimana kedua nilai yang berdekatan hanya mempunyai perbedaan satu digit [3]. Penggunaan kode Gray akan membantu mengurangi bit error yang terjadi. Jumlah titik-titik pada diagram berupa pemangkatan dari dua (2^n) karena pada komunikasi digital, datanya berbentuk biner.

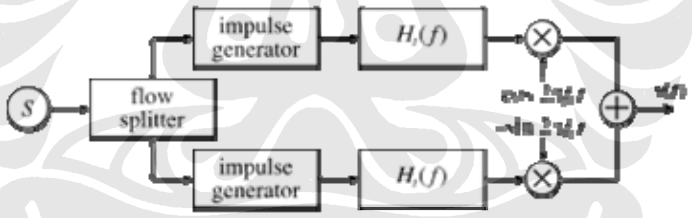


Gambar 2.2 Diagram konstelasi sistem 16-QAM [3]

Sumbu x merupakan sumbu yang mewakili $\cos(2\pi f_0 t)$ dari persamaan 2.1 dan disebut sebagai sumbu I (*inphase*), sedangkan sumbu y adalah sumbu yang mewakili $\sin(2\pi f_0 t)$ dari persamaan 2.1 dan disebut sebagai sumbu Q (*quadrature*).

2.1.2 QAM *Transmitter*

Pada QAM *transmitter*, aliran data yang masuk dibagi menjadi kumpulan bit kemudian dikonversi menjadi suatu simbol. Kemudian, sinyal informasi ini dibagi menjadi dua yaitu komponen *inphase* dan komponen *quadrature,* lalu difilter. Selanjutnya, komponen *inphase* dikalikan dengan sinyal kosinus, sedangkan komponen *quadrature* dikalikan dengan sinyal sinus. Hal ini sesuai dengan prinsip QAM yaitu beda fase antara kedua *carrier* sebesar 90^0. Kemudian kedua komponen ini dijumlahkan dan siap untuk ditransmisikan. Blok diagram QAM *transmitter* ini dapat dilihat pada Gambar 2.3.



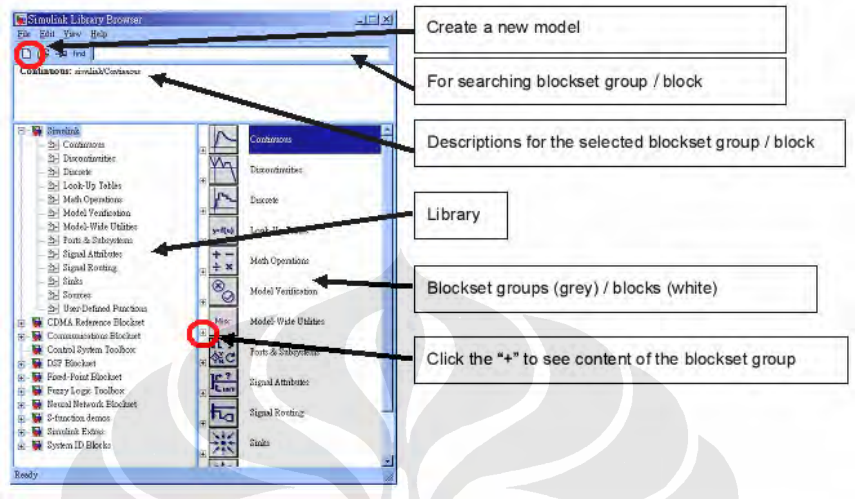
Gambar 2.3 Blok diagram QAM *transmitter* [4]

2.2 SIMULINK

Simulink adalah sebuah *software* yang terintegrasi dalam MATLAB, yang mampu memprogram secara visual sebuah sistem dinamis dan langsung melihat hasilnya secara *real-time* [5]. Rangkaian logika atau sistem kendali sembarang dapat dibentuk dengan menggunakan *building block* standar yang tersedia pada pustaka Simulink. Tampilan dari Simulink *library browser* ini dapat dilihat pada Gambar 2.4.

Macam-macam *toolbox* yang terdapat pada *library browser* dengan teknik yang berbeda seperti *fuzzy logic, neural network,* DSP, statistika, dan sebagainya

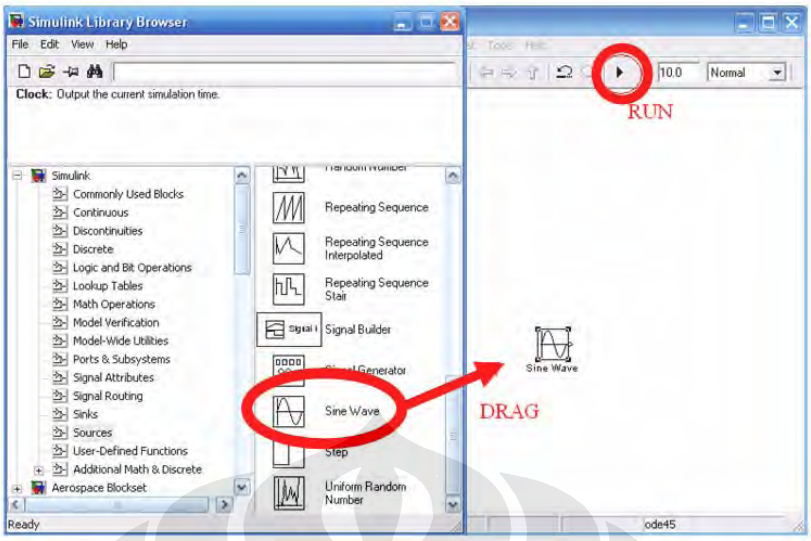
tersedia pada Simulink. Keuntungan yang paling utama dari ketersediaan *template/building block*, yaitu menghindari pengetikan kode yang kompleks yang diperlukan untuk berbagai proses matematika sehingga sangat memudahkan *user*.



Gambar 2.4 Simulink *library browser* [6]

2.2.1 Konsep Dasar Simulink

Dalam Simulink, data/sinyal dari bermacam-macam blok dikirimkan ke blok lain dengan menghubungkan garis ke blok yang bersangkutan. Sinyal dapat dibangkitkan atau diberikan ke dalam blok (dinamis/statis), atau diberikan ke dalam suatu fungsi. Data dapat ditampung di *sink*, yang dapat berupa osiloskop virtual, *display*, atau disimpan ke dalam file. Dalam simulasi, data diproses dan dikirimkan hanya pada waktu diskrit, karena semua komputer adalah sistem diskrit.



Gambar 2.5 Proses *drag* dan *run* pada Simulink

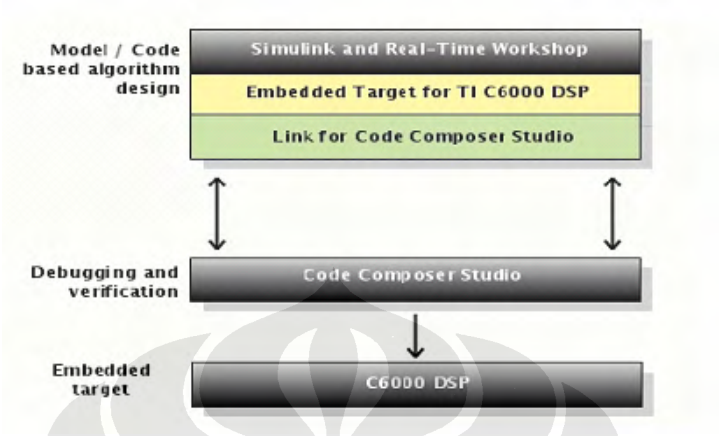
Langkah awal dalam pembuatan model yaitu memasukkan blok yang ingin dipakai kedalam jendela model. Caranya yaitu dengan memilih blok yang diinginkan dari Simulink *library*, klik kiri mouse, kemudian tahan sambil men-*drag* blok tersebut ke jendela model. Untuk menghubungkan blok, klik-kiri *mouse* dari output suatu blok kemudian *drag* menuju input dari blok selanjutnya,maka blok tersebut akan terhubung dengan sebuah garis panah. Proses *drag* dan *run* pada Simulink dapat dilihat pada Gambar 2.6.

Setelah model yang diinginkan telah terbentuk, simulasi dapat dijalankan dengan cara menekan tombol *run* Hasil simulasi dapat dilihat dalam sebuah *sink*, atau pada *workspace*. S*ink* adalah blok dimana sinyal akan diakhiri atau ditampilkan hasilnya, contohnya yaitu *scope*, *diagram scatter plot scope* dan lain-lain.

2.2.2 *Link to DSP C6000*

The Embedded Target for TI C6000 DSP menyediakan *Application Programming Interface* (API) yang dibutuhkan oleh *real time workshop* untuk membangkitkan kode secara khusus untuk C6000 DSP. Simulink menggunakan pendekatan *block based implementation* sebagai ganti dari pembuatan algoritma kompleks secara manual. *Link for Code Composer Studio* (CCS) diperlukan untuk proses pembuatan kode. Kode yang dibangkitkan kemudian di*download* ke DSP

target melalui perantara CCS. Diagram alir proses sinkronisasi antara Simulink dengan DSP C6000 dapat dilihat pada Gambar 2.7.



Gambar 2.6 Diagram alir Simulink dengan DSP C6000 [7]

Pustaka blok yang digunakan untuk proses pemrograman TI C6713DSK dapat dilihat pada Gambar akan dijelaskan sebagai berikut:

1. Blok C6713 DSK ADC

   Blok ini digunakan untuk menangkap dan mendigitalisasikan sinyal *analog* dari sumber luar, seperti sinyal generator, frekuensi generator, atau alat audio. Dengan memakai C6713 DSK ADC blok pada model Simulink akan membuat audio *coder-decoder module* (codec) pada DSK C6713 mengubah masukan sinyal *analog* menjadi sinyal digital untuk pemrosesan sinyal digital.

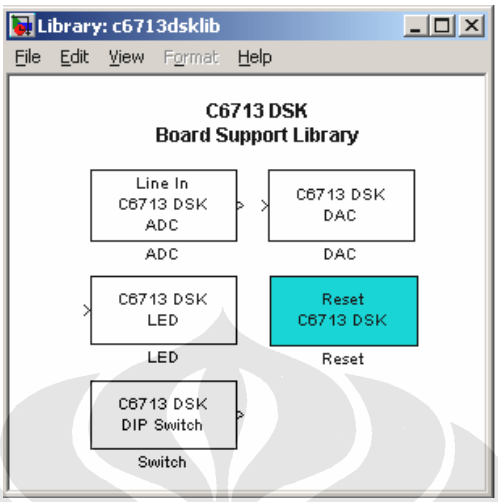
2. Blok C6713 DSK DAC

   Pemodelan Simulink mampu membangkitkan keluaran berupa sinyal *analog* menuju ke *jack* keluaran *analog* pada DSK C6713. Ketika DSK C6713 DAC blok ditambahkan ke dalam model, sinyal digital yang diterima *codec* dikonversi menjadi sinyal *analog*. Codec mengirimkan sinyal ini ke *jack* keluaran setelah melakukan konversi *digital-to-analog* (D/A).

3. Blok C6713 DSK *Target Preferences*

   Blok ini memberikan akses ke pengaturan *hardware processor* yang perlu dikonfigurasi untuk membangkitkan kode dari *Real-Time Workshop* untuk menjalankan target. Penambahan blok ini ke dalam model harus dilakukan jika ingin menggunakan suatu DSK melalui Simulink. Blok ini berlokasi pada

Target Preference di bagian *Embedded Target for* TI C6000 DSP untuk pustaka TI.



Gambar 2.7 Pustaka C6713DSK

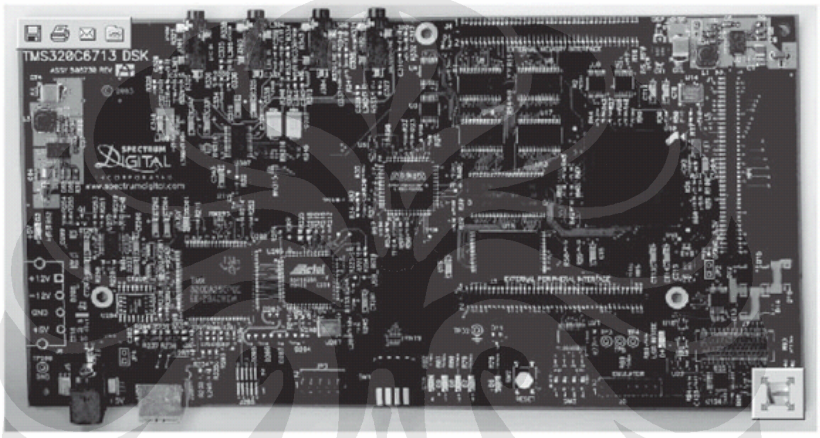
2.3 DSP Starter Kit (DSK) TMS320C6713

Digital Signal Processing (DSP) *processor*, seperti keluarga *processor* TMS320C6x adalah *mikroprocessor* berkecepatan tinggi dengan tipe arsitektur dan set instruksi khusus untuk pemrosesan sinyal. Notasi C6x menandakan bahwa *processor* tersebut anggota dari Texas Instrument (TI) keluarga *processor* TMS320C6000. Arsitektur dari C6x digital signal *processor* dikhususkan untuk perhitungan numerik yang sangat kompleks. Berdasarkan arsitektur *very-long-instruction-word* (VLIW), C6x dianggap sebagai *processor* TI yang terbaik dibanding lainnya [8].

DSP *processor* berkaitan erat dengan pemrosesan sinyal secara *real-time*. Berbagai teknologi telah digunakan untuk proses *real-time*, mulai dari fiber optik untuk frekuensi yang sangat tinggi sampai DSP *processor* yang bekerja pada rentang frekuensi audio. Sistem dasar DSP *processor* terdiri dari *analog-to-digital converter* (ADC) untuk menangkap sinyal masukan *analog*. Hasil representasi digital dari sinyal kemudian diproses oleh DSP *processor*, misal C6x, kemudian hasil keluarannya diubah kembali menjadi sinyal *analog* melalui *digital-to-analog converter* (DAC).

Paket DSP Starter Kit (DSK) TMS320C6713 terdiri dari beberapa alat, yaitu antara lain:

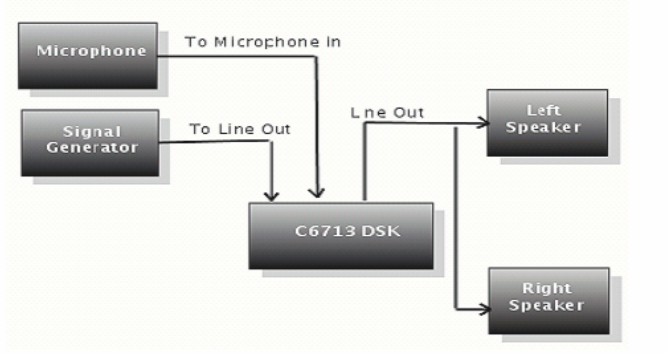
1. Code Composer Studio (CCS), yang merupakan software pendukung yang diperlukan untuk pemrograman DSP *processor*.
2. Sebuah *board*, dapat dilihat pada Gambar 2.8, yang terdiri dari DSP *processor* TMS320C6713 dan codec 32-bit stereo untuk dukungan input dan output (I/O).
3. Kabel *universal synchronous bus* (USB) untuk koneksi DSK *board* ke komputer.
4. *Power supply* 5V untuk DSK *board*.



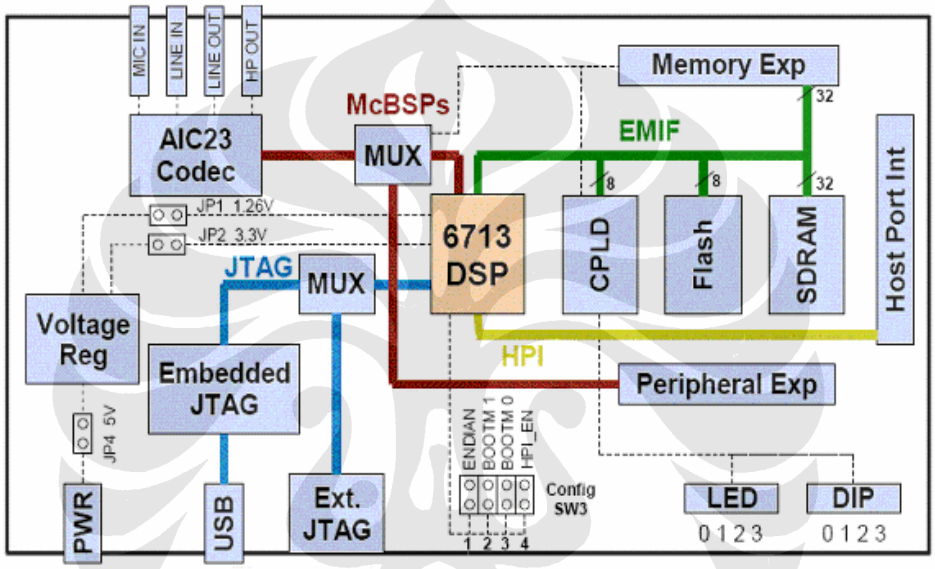
Gambar 2.8 TMS320C6713 DSK *board* [8]

2.3.1 DSK *board*

DSK *board*, yang berukuran kurang lebih 5 x 8 inchi, terdiri dari DSP *processor* C6713 dan *codec* 32-bit stereo AIC23 untuk input dan output yang terdiri dari ADC dan DAC. Selain itu, terdapat 16 MB SDRAM dan 256 kB *flash* memori. DSK beroperasi pada frekuensi 225 MHz. Pada DSK *board* terdapat pula regulator tegangan yang memberikan 1.26 V untuk C6713 *processor* dan 3.3 V untuk memori dan yang lainnya. Empat konektor pada *board* tersedia untuk input dan output: *MIC IN* untuk input *microphone*, *LINE IN* untuk saluran input, *LINE OUT* untuk saluran output, dan *HEADPHONE* untuk output *headphon*e. Empat konektor ini dapat dilihat pada Gambar 2.9.



Gambar 2.9 C6713 input-ouput *jack* [7]



Gambar 2.10 Diagram DSK TMS 320C6713 [9]

2.3.2 TMS320C6713 *Digital Signal Processor*

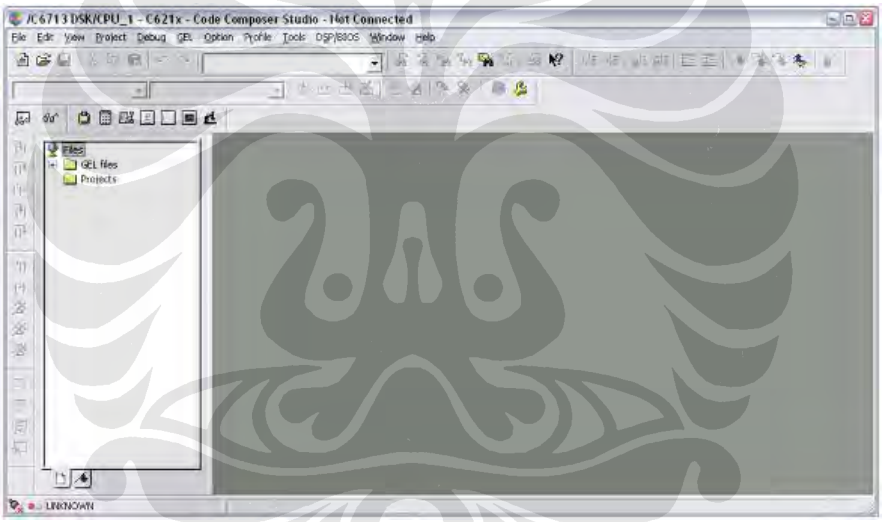
DSP *processor* C6713 dibuat berdasarkan arsitektur *very-long-instruction-word* (VLIW), yang sangat cocok untuk algoritma numerik kompleks [8]. DSP *processor* C6713 ini berada di tengah DSK *board*, dapat dilihat pada Gambar 2.10. Memori program internal yang ada mampu melakukan 8 instruksi tiap *cycle*. Misalnya, untuk *clock rate* sebesar 225 Mhz, C6713 mampu menampung delapan instruksi 32-bit tiap 1/(225 MHz) atau sebesar 4,44 ns.

Keutamaan C6713 antara lain 246 kB memori internal, delapan unit fungsional yang disusun dari enam arithmatic-logic unit (ALU) dan dua unit multiplier, bus alamat 32 bit sampai 4 GB (*gigabyte*), dan dua set 32-bit *general-purpose register*. DSP *processor* C67xx (seperti C6701,C6711, C6713) termasuk

dalam keluarga *processor* C6x *floating-point*, sedangkan C62xx dan C64xx termasuk dalam keluarga C6x *processor fixed-point*. C6713 *processor* mampu melakukan proses baik *fixed* maupun *floating-point*.

2.4 CODE COMPOSER STUDIO

Code Composer Studio (CCS) merupakan *software integrated development environment* (IDE) yang merupakan gabungan dari C *compiler*, *assembler*, *linker*, dan *debugger* terintegrasi [8]. CCS merupakan software *easy-to-use* untuk *build* dan *debug* program pada DSP *processor*, dan mampu melakukan analisis secara real time. CCS diperlukan oleh komputer untuk proses *compile* dan *download* kode serta menjalankannya di *processor*. Tampilan *window* dari CCS dapat dilihat pada Gambar 2.11.



Gambar 2.11 Tampilan dari CCS

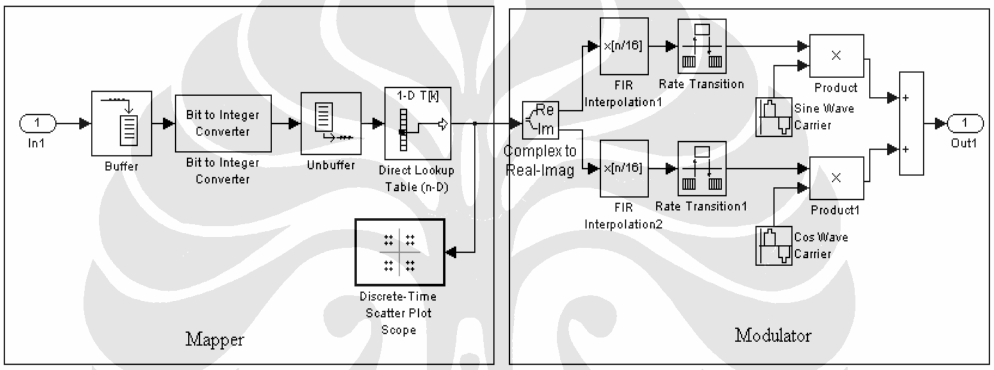
C *compiler* membentuk program berbahasa C dengan ekstensi .c untuk menghasilkan file bahasa mesin dengan ekstensi ".asm". Kemudian *assembler* mengubahnya menjadi file objek bahasa mesin dengan ekstensi ".obj". *Linker* akan menggabungkan file objek dengan pustaka objek sebagai input untuk menghasilkan *executable file* dengan ekstensi ".out". Executable file ini merepresentasikan format *file linked common object* (COFF), yang kemudian dapat dimuat dan dijalankan secara langsung oleh *processor* C6713 [8].

BAB III

RANCANG BANGUN SISTEM 16-QAM

3.1 RANCANG BANGUN 16-QAM MENGGUNAKAN SIMULINK

Rancang bangun sistem 16-QAM dilakukan dengan menggunakan program Simulink yang terdapat pada *software* MATLAB. Model sistem 16-QAM yang telah dibuat dapat dilihat pada Gambar 3.1. Sumber sinyal yang digunakan dalam model adalah blok *bernoulli binary generator*, yang akan membangkitkan sinyal diskrit (bit 0 dan 1).



Gambar 3.1 Model sistem 16-QAM

Secara umum, terdapat dua proses yang dilakukan pada simulasi ini yaitu pemetaan sinyal dan penumpangan sinyal carrier pada sinyal asli. Berikut ini penjelasan singkat dari kedua proses tersebut:

1. *Mapper* atau pemetaan, sistem ini melakukan proses pemetaan sinyal ke dalam kode gray dan mengubah amplitudo dan fasenya. Dalam subsistem ini bit-bit input di*buffer* tiap 4 bit kemudian diubah menjadi sebuah integer, kemudian nilai ini di*unbuffer* dan dipetakan dengan menggunakan *lookup table*. Hasilnya dapat dilihat pada *discrete-time scatter plot scope* yang merepresentasikan diagram konstelasi 16-QAM.
2. Modulator, sistem ini melakukan proses penumpangan sinyal *carrier* ke dalam sinyal asli. Disini sinyal akan dipecah menjadi dua yaitu menjadi komponen *Inphase* dan *Quadrature*. Kemudian dilakukan proses interpolasi dan pemfilteran sinyal. Setelah itu, sinyal asli ini dikalikan dengan sinyal *carrier* yaitu berupa gelombang sinus dan kosinus lalu dijumlahkan.

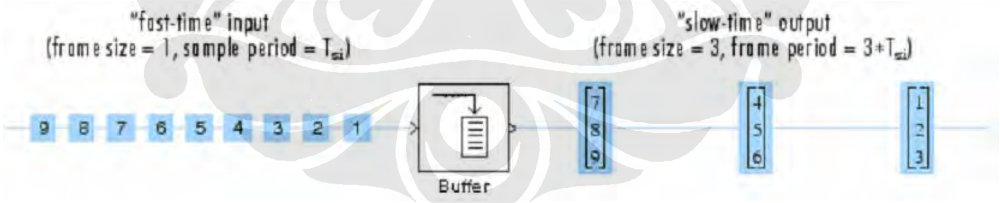
Sinyal keluaran ini kemudian ditampilkan ke dalam sebuah *scope* untuk dilihat hasilnya. Sinyal inilah yang merupakan hasil dari modulasi 16-QAM. Pada subbab berikutnya akan dijelaskan secara umum kegunaan tiap blok yang digunakan pada model simulasi.

3.1.1 *Bernoulli Binary Generator*

Blok ini berfungsi untuk membangkitkan sejumlah nilai biner yang acak dengan menggunakan distribusi Bernoulli. Distribusi Bernoulli dengan parameter p, menghasilkan nilai nol dengan probabilitas p dan nilai 1 dengan probabilitas 1-p. Pada simulasi ini parameter p diset menjadi 0,5 dengan sample time sebesar 1 detik.

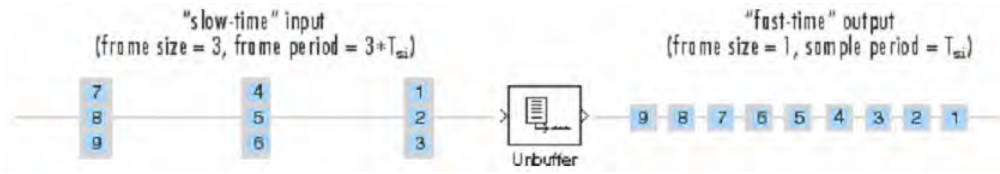
3.1.2 *Buffer* Dan *Unbuffer*

Buffer berfungsi untuk mengubah sequence input sample skalar menjadi frame output dengan *rate* yang lebih rendah *Buffer* mengubah ukuran frame sample input ke *frame size* yang baru menjadi lebih besar atau lebih kecil. Mem*buffer* sinyal ke *frame size* yang lebih besar akan membuat output yang dihasilkan *frame rate* yang lebih rendah, seperti ilustrasi dibawah ini. Blok ini berfungsi untuk mengubah inputan *frame* menjadi *sample* skalar pada *rate* yang lebih tinggi.



Gambar 3.2 Proses *buffer* pada sistem [10]

Sedangkan *unbuffer* berfungsi untuk mengubah inputan *frame* menjadi *sample* skalar pada *rate* yang lebih tinggi. Sinyal ini akan mempunyai *rate* yang sama seperti sebelum di*buffer*.



Gambar 3.3 Proses *unbuffer* pada sistem [10]

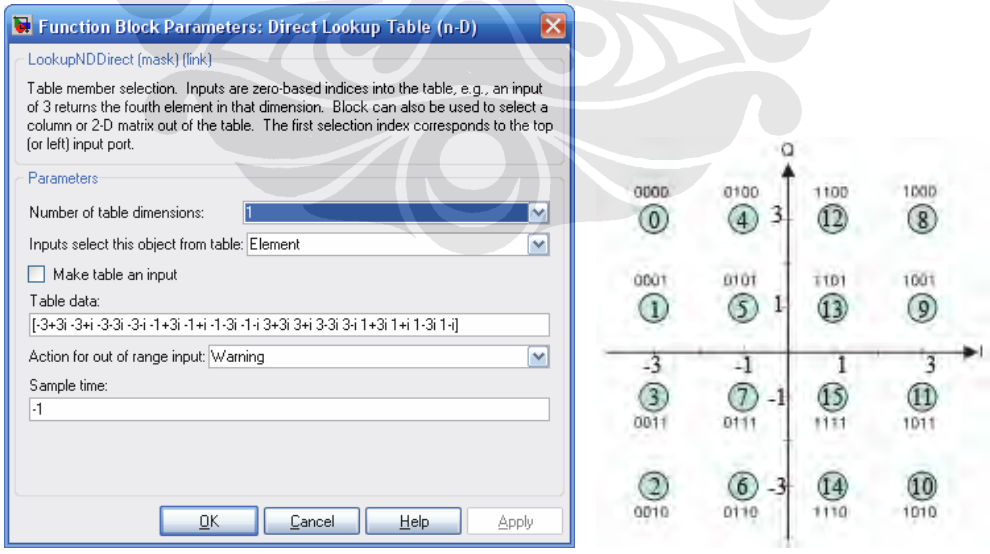
Pada simulasi ini output *buffer* size diset bernilai 4, sehingga *frame size*nya menjadi 4x1, setelah di*unbuffer* maka maka *frame size* kembali menjadi 1. Tujuan dilakukan *buffer* adalah untuk mengkonversi tiap nilai 4 bit menjadi satu simbol integer. Hal ini sesuai dengan prinsip 16-QAM (2^4 bit) yaitu dalam satu simbol akan terdapat 4 bit.

3.1.3 Bit To Integer Converter

Bit to Integer converter blok memetakan kumpulan bit dalam masukan vektor menjadi integer. Jika M adalah jumlah bit per integer parameter maka block akan memetakan tiap M-grup bit menjadi integer bernilai antara 0 sampai 2^M-1. Pada simulasi ini, *number of bits per integer* diset menjadi 4, artinya tiap 4 bit akan dikonversi menjadi sebuah nilai integer.

3.1.4 Direct Lookup Table

Blok *Direct lookup table* (n-D) akan memetakan inputannya sebagai indeks nol lalu mengubahnya menjadi tabel n dimensi. Jumlah input beragam bergantung dari bentuk output yang diinginkan. Outputnya dapat berupa satu elemen, kolom, atau 2-D matriks. Fungsi umum *lookup table* dalam simulasi ini adalah memetakan sinyal ke dalam komponen *inphase* dan *quadrature*



Gambar 3.4 Konfigurasi parameter *Lookup Table* dan diagram konstelasinya.

Pada rancangan ini dimensi tabel diset menjadi 1 sesuai dengan inputnya. Kemudian nilai integer hasil konversi yaitu dari 1 sampai 16 dipetakan dengan *lookup table* dengan indeks 0 sampai 15 dengan koordinat masing-masing 3, -1, 1, 3 pada sumbu real maupun imajiner mengikuti aturan 16-QAM. Sebagai contoh untuk indeks ke nol akan dipetakan ke -3 pada sumbu real dan +3i pada sumbu imajiner. Diagram konstelasi dari pemetaan beserta parameternya pada simulasi ini dapat dilihat pada Gambar 3.4.

3.1.4 *Discrete-time Scatter Plot*

Discrete-time scatter plot mempunyai satu port input dan inputnya harus dalam bentuk kompleks. Dalam simulasi ini digunakan untuk melihat hasil dari proses pemetaan yang dilakukan dan menganalisa hasilnya.

3.1.6 Complex To Real-Imag

Blok ini berfungsi untuk mengkonversi sinyal kompleks menjadi real dan imajiner pada outputnya. Dalam simulasi ini berfungsi memecah sinyal menjadi komponen *inphase* dan *quadrature* untuk proses perkalian dengan sinyal *carrier* masing-masing.

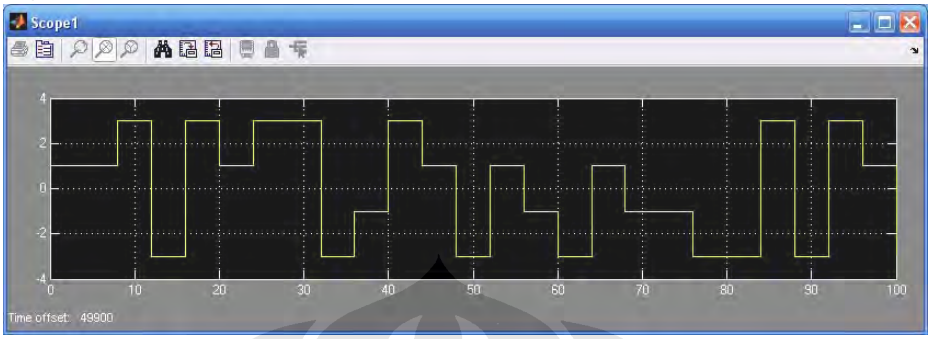
3.1.7 FIR Interpolation

Blok ini berfungsi untuk meningkatkan *sample rate* input diskrit menjadi L kali lebih cepat dibanding sebelumnya sekaligus berfungsi sebagai *lowpass* filter. Nilai L ditentukan oleh parameter *interpolation factor.* Proses ini terdiri dari dua langkah:

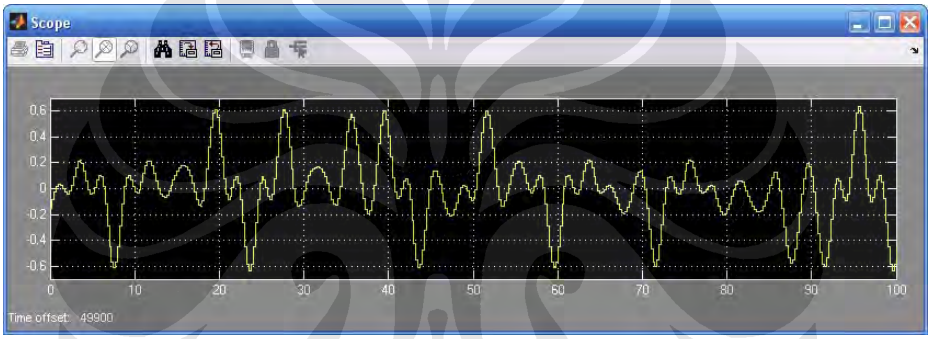
1. *Sample time* input ditingkatkan dengan cara menyisipkan L-1 zero diantara sample.
2. Data kemudian difilter dengan *direct-form FIR filter*. Filter ini akan berfungsi sebagai *lowpass* filter dengan frekuensi *cut-off* yang dinormalisasi tidak lebih besar dari 1/L.

Hasil dari proses FIR *interpolation* dapat dilihat pada Gambar 3.5. Koefisien FIR filter diset menjadi fir(60,0,2) yang artinya filter menggunakan orde 60 dan frekuensi *cut-off* normalisasinya sebesar 0.2. Nilai *cut-off* ini mempunyai ketentuan yaitu harus lebih kecil atau sama dengan 1/L. *Interpolation*

factor (L) yang digunakan adalah sebesar 16, sehingga frekuensi *cut-off*nya harus lebih kecil dari 1/16 (0.0625).



(a)



(b)

Gambar 3.5 FIR *interpolation*. (a) sebelum. (b) sesudah

3.1.8 *Rate Transition*

Blok ini berfungsi untuk mentransfer data dengan *rate* yang berbeda antara input dan output. Untuk menyamakan *rate* antara input dan output *rate transition* dapat berperan sebagai *zero order hold, unit delay, buffer* dan lain-lain. Dalam simulasi ini *rate transition* berfungsi sebagai *unit delay* dengan lambang 1/z.

3.1.9 *Sine Wave*

Blok ini berfungsi untuk membangkitkan gelombang sinusoidal. Blok ini dapat beroperasi baik dalam mode *time-based* maupun *sample-based*. Dalam perancangan ini digunakan mode *time-based*. Output dari *sine wave* blok ditentukan oleh persamaan 3.1.

$$y = amplitude \times \sin\left(frequency \times time + phase \right) + bias \quad \text{.....(3.1)}$$

Time based mode dibagi lagi menjadi dua, yaitu *continous mode* dan *discrete mode*. Blok ini digunakan untuk membangkitkan sinyal *carrier*. Untuk perancangan ini dipilih mode diskrit, dengan parameter amplitude diset bernilai 5, frekuensi diset menjadi $2\pi/ 0.5$ dan sample time 1/100, jadi frekuensi yang digunakan sebesar 2 Hz. Nilai frekuensi ini dipilih hanya sebagai nilai perbandingan dengan frekuensi asli yaitu diatas 2400 Hz, karena penggunaan frekuensi yang sangat tinggi pada simulasi dapat menyebabkan program *hang*. Phase diset 0 untuk komponen *inphase*, sedangkan untuk komponen *quadrature* diset mejadi 90. Hal ini sesuai dengan prinsip dari QAM yaitu beda fase antara kedua sinyal *carrier* sebesar 90^0 sehingga seolah-olah menjumlahkan sebuah sinyal AM yang menggunakan *carrier* kosinus dengan sebuah sinyal AM lain yang menggunakan *carrier* sinus.

3.1.10 *Product*

Blok ini berfungsi untuk mengalikan sinyal-sinyal inputnya. Pada simulasi ini berfungsi untuk mengalikan sinyal asli, baik pada komponen *inphase* maupun *quadrature*, yang telah diinterpolasi dengan sinyal *carrier*.

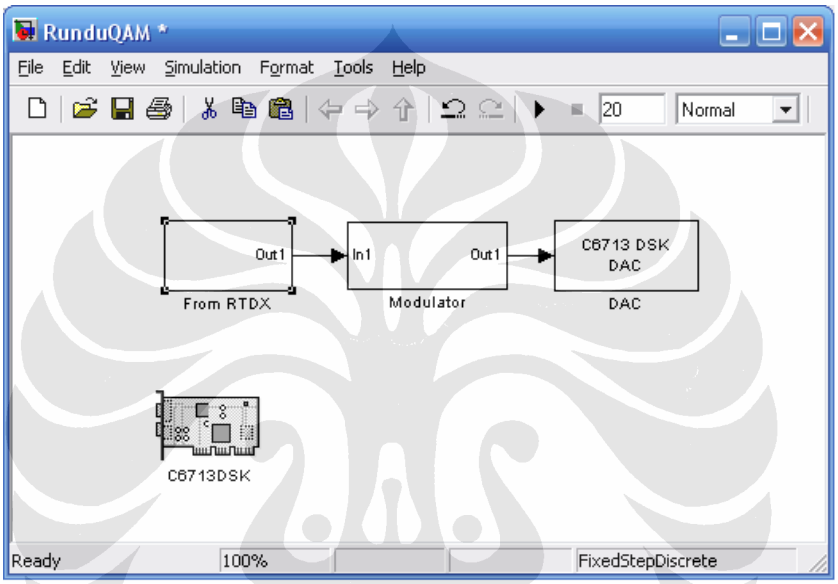
3.1.11 *Sum*

Blok ini digunakan untuk menjumlahkan atau mengurangkan sinyal-sinyal inputnya. Blok ini dapat menjumlahkan input berupa scalar, vektor atau matriks. Penentuan operasi dilakukan melalui pemilihan parameter *List of sign.* Karakter *plus* (+), *minus* (-), and *spacer* (|) menyatakan operasi yang dilakukan pada input blok ini. Pada simulasi ini blok ini berfungsi untuk menjumlahkan komponen *inphase* dan *quadrature* yang telah dikalikan dengan sinyal *carrier*.

3.2 RANCANG BANGUN SISTEM MENGGUNAKAN DSK TMS320C6713

Rancang bangun sistem 16-QAM pada DSK ini dilakukan melalui program Simulink tanpa perlu melakukan pemrograman manual yang rumit. Dengan menggunakan *Link to* C6000 DSP yang tersedia pada pustaka Simulink, model sistem yang telah dibuat dapat dibangkitkan kode programnya secara otomatis. Sebelum program di*build*, model sistem yang telah dibuat harus dipastikan dalam keadaan diskrit.

Dalam perancangan ini, pada input modulator ditambahkan *From* RTDX, sedangkan output akhir dari sistem dihubungkan dengan blok C6713DSK DAC. Selain itu perlu ditambahkan blok C6713DSK *target preference* pada model sistem agar kode program dapat dibangkitkan dari *Real Time Workshop* dan dijalankan pada DSK. *Simulation time* diset pada detik ke-20 dan ke-30 untuk dua kali pengambilan data. Model sistem 16-QAM yang telah diubah dan siap di*build* ke dalam DSK TMS320C6713 dapat dilihat pada Gambar 3.6



Gambar 3.6 Model sistem 16-QAM yang akan dibangun

Sebagai sumber dipakai blok *From* RTDX yang berfungsi untuk mendapatkan data dalam bentuk *frame* dari komputer melalui fasilitas RTDX. Data yang diambil dari komputer merupakan data yang dibangkitkan oleh *Bernoulli Binary Generator* untuk detik ke-20 dan ke-30 yang telah sebelumnya disimpan pada file .mat. Pada proses selanjutnya, data *frame* tersebut dikonversi menjadi ukuran skalar dengan menggunakan *unbuffer*. Kemudian aliran data ini akan masuk ke dalam blok modulator untuk diproses lebih lanjut. File ”RTDXdriver.m” yang dapat dilihat pada bagian lampiran digunakan untuk mengendalikan proses transfer data antara komputer dengan C6713 DSK.

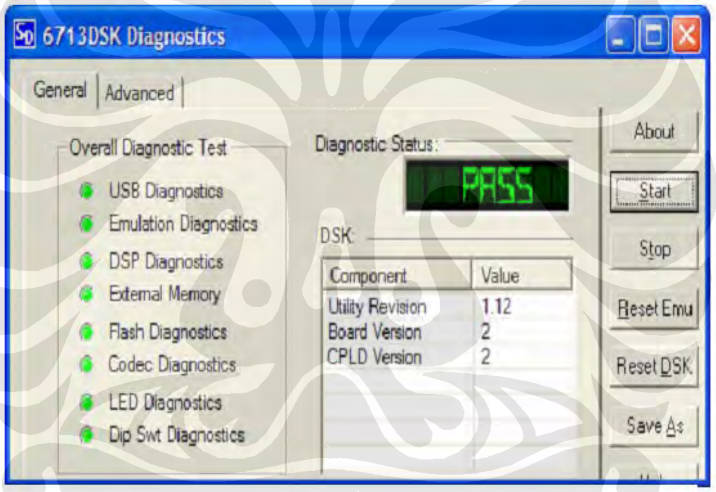
Setelah data hasil simulasi yang didapat sesuai dengan teori, maka blok *From* RTDX dapat diganti dengan blok C6713DSK ADC. Input nantinya didapat dari line in DSK, sehingga DSK akan menjadi modulator 16-QAM yang

sebenarnya. Pemakaian *From* RTDX pada Gambar 2.6 hanya digunakan untuk mempermudah analisis perbandingan data hasil rancang bangun pada Simulink dengan DSK TMS320C6713. Karena jika digunakan sinyal analog langsung menuju line in DSK, akan sulit dilakukan perbandingan data dengan ouputnya.

3.2.1 Konfigurasi Awal

Untuk mengecek *software* CCS terinstal dengan baik dan DSK telah siap untuk diprogram, maka perlu dilakukan konfigurasi awal sebelum dilakukan perancangan. Berikut ini beberapa hal yang perlu dilakukan **:**

1. Pada CCS, *diagnostic test* dilakukan untuk memastikan *board* telah terhubung dengan baik dan tidak mengalami kerusakan. Tes ini dapat dilihat pada Gambar 3.7.



Gambar 3.7 CCS diagnostic test

2. Untuk memastikan CCS terinstal dengan baik pada komputer, pada MATLAB *command* ketikkan ”*ccsboardinfo*”. MATLAB akan menampilkan informasi seperti pada Gambar 3.8.

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

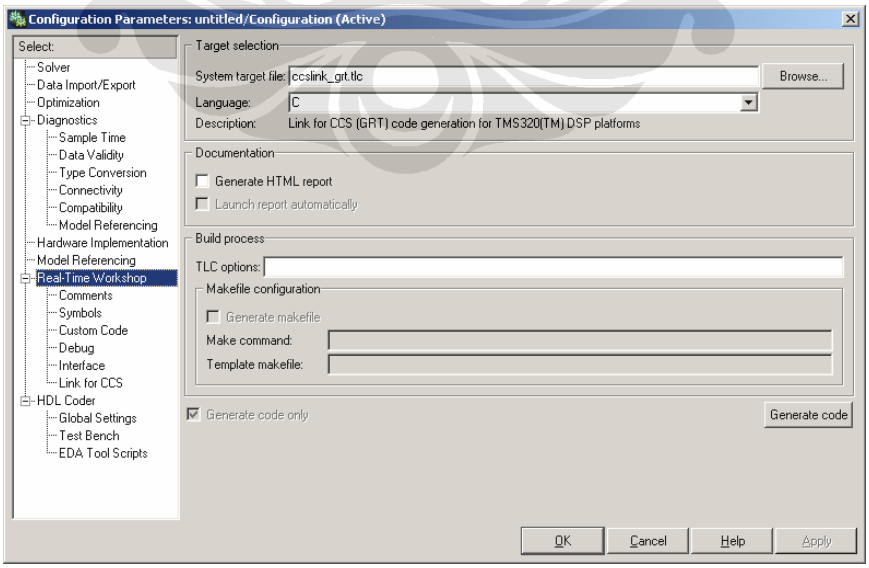
Gambar 3.8 Informasi cc*sboard*info pada MATLAB

3. Untuk memastikan *Embedded Target for* TI C6000 DSP terinstal, ketikkan ”c6000lib” pada MATLAB *command.* MATLAB akan menampilkan C6000 blok *library*.

3.2.2 *Configuration Parameter* Untuk C6000 *Hardware*

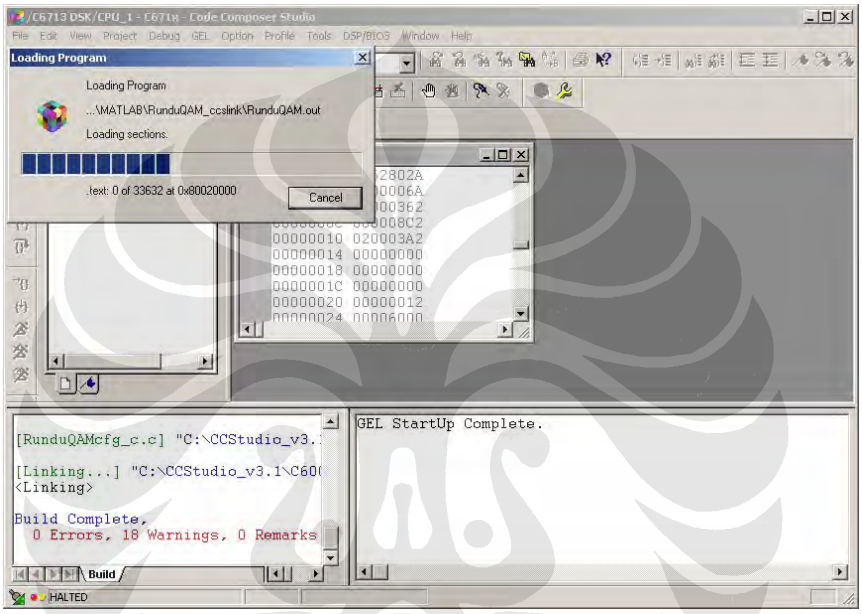
Selain itu, perlu dilakukan pengaturan Simulink dan *Real Time Workshop* agar model ini dapat kita build pada C6000 DSP. Pengaturan konfigurasi ini diperlukan dan pengaturannya sama untuk semua eksperimen. Parameter ini dapat kita akses pada toolbar *Simulation* → *Configuration Parameter*. Berikut ini beberapa hal yang perlu kita atur :

1. Pada *Real Time Workshop* kategori, pada *Target selection*, pilih file ccslink_grt.tlc. Hal ini dapat dilihat pada Gambar 3.9.
2. Pada *Link for CCS*, pada bagian *Project option*, set nilai *System stack size* menjadi 1024 byte.
3. Pada *Debug*, pilih *Verbose build mode.*
4. Pada *Optimization* kategori, untuk *Simulation and Code generation unselect Block reduction optimization and Implement logic signal as boolean data.*
5. Pada *Solver* kategori, pastikan *solver* yang dipilih adalah *Fixed type dan discrete.*



Gambar 3.9 Real Time Workshop configuration parameter

Setelah konfigurasi diatas telah selesai dilakukan, selanjutnya tinggal menekan *icon incremental build*. Jika tidak terjadi error, maka Simulink akan membangkitkan kode program secara otomatis. Sistem akan meng-*compile* kode pada Matlab, kemudian *Code Composer Studio* akan terbuka, lalu *assembly code* akan di*compile* dan dimuat ke dalam DSP. Proses *load* ini dapat dilihat pada Gambar 3.10.



Gambar 3.10 Program dimuat ke DSK *board*

Program yang berekstensi .out yang telah di*compile* akan langsung *running* begitu telah selesai dimuat ke dalam DSK *board* dan akan terus berjalan sampai ada perintah untuk berhenti. Untuk menghentikan program yang sedang berjalan, gunakan perintah *Halt* pada CCS.

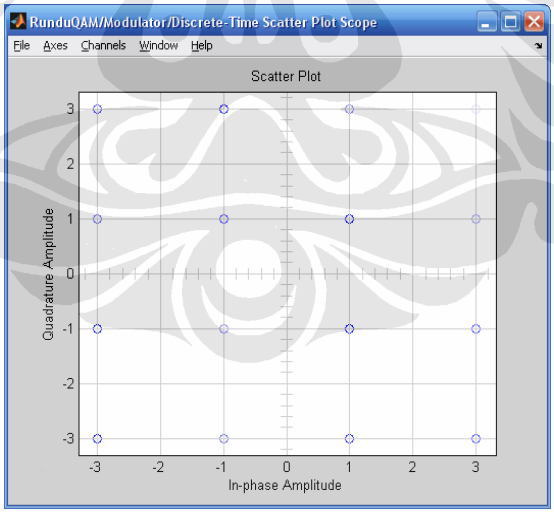
Setelah memastikan kode yang telah dimuat berjalan, maka dilakukan pengambilan data dengan menggunakan TDS3000B *Oscilloscope*. Osiloskop ini dihubungkan pada *lineout* DSK TMS320C6713, kemudian data yang didapat dibandingkan dengan hasil output pada Simulink.

BAB IV
UJI COBA DAN ANALISIS

4.1 UJI COBA RANCANG BANGUN PADA SIMULINK

Analisis rancang bangun 16-QAM pada Simulink dapat diperoleh dengan melakukan uji coba dan mengamati hasil output dari dua *scope* yang ada pada model, yaitu *discrete time scatter plot scope* dan *scope* pada output akhir. *Discrete time scatter plot* ini merepresentasikan diagram konstelasi sistem hasil keluaran dari mapper, sedangkan *scope* berikutnya menampilkan output setelah sinyal ditumpangi dengan carrier.

Dari hasil output *discrete time scatter plot*, telihat bahwa pada detik pertama hingga detik ke delapan, hanya muncul sebuah titik yaitu pada titik pusat (0,0). Kemudian pada detik selanjutnya, barulah muncul 16 titik tersusun rectangular pada keempat kuadran yang polanya mengikuti aturan 16-QAM. Output dari *discrete time scatter plot* yang berupa diagram konstelasi hasil keluaran mapper dapat dilihat pada Gambar 4.1.

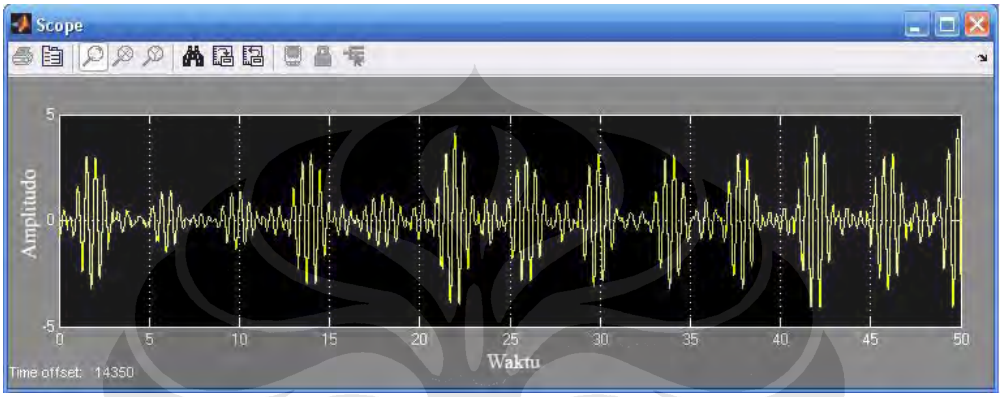


Gambar 4.1 Keluaran dari *discrete time scatter plot*

Delay yang terjadi pada sistem selama 8 detik terjadi karena penggunaan *buffer* dan *unbuffer* pada model. Buffer digunakan untuk menahan data sebanyak 4 bit yang nantinya akan dikonversi oleh *bit to integer converter* untuk menghasilkan satu simbol integer, lalu kemudian data di-*unbuffer*. Karena sample

time tiap bit sebesar 1 detik, maka akan terdapat *delay* sebesar 8 detik untuk proses *buffer* dan *unbuffer*.

Dari output akhir sistem pada Gambar 4.2, didapatkan sinyal berbentuk sinusoidal dengan amplitudo dan fasa yang berbeda-beda sesuai dengan persamaaan 2.1 yang berupa hasil penjumlahan komponen *inphase* dan *quadrature* yang telah dikalikan dengan sinyal carrier.



Gambar 4.2 Keluaran akhir dari sistem 16-QAM

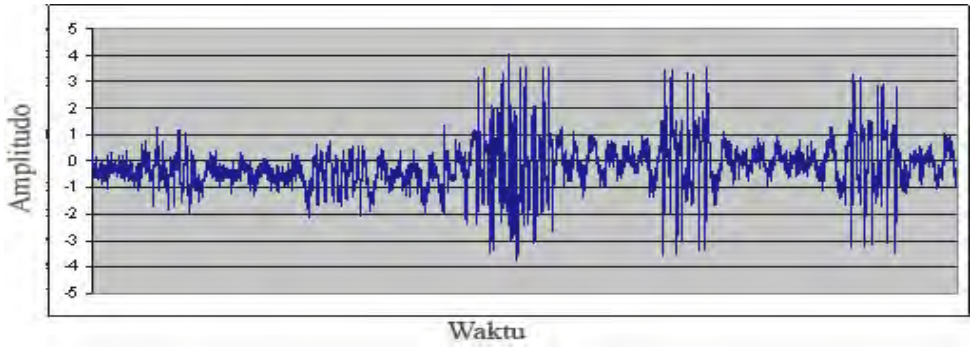
Dari analisis hasil kedua *scope* diatas, dapat disimpulkan bahwa rancang bangun 16-QAM pada Simulink yang dilakukan telah sesuai dengan teori.

4.2 UJI COBA RANCANG BANGUN PADA DSK TMS320C6713

Setelah program 16-QAM dimuat ke dalam DSP *processor* dan berjalan, maka dilakukan pengambilan data untuk menganalisis hasil dari rancang bangun yang telah dibuat. Pengambilan data dilakukan dengan menggunakan *digital storage oscilloscope* yang dihubungkan dengan *lineout* dari DSK TMS320C6713.

Dengan menggunakan osiloskop ini, sinyal yang tampil pada layar *osiloskop* dapat disimpan dalam sebuah disket dengan berbagai macam format. Pada pengujian, data yang diambil berupa sebuah *file* yang berekstensi .csv yang berukuran sekitar 160 KB dan dapat kita buka dengan menggunakan Microsoft Excel. Secara *default*, maka akan ada 10.000 data dalam file ini, karena dalam tampilan layar osiloskop suatu signal akan direpresentasikan oleh 10.000 titik.

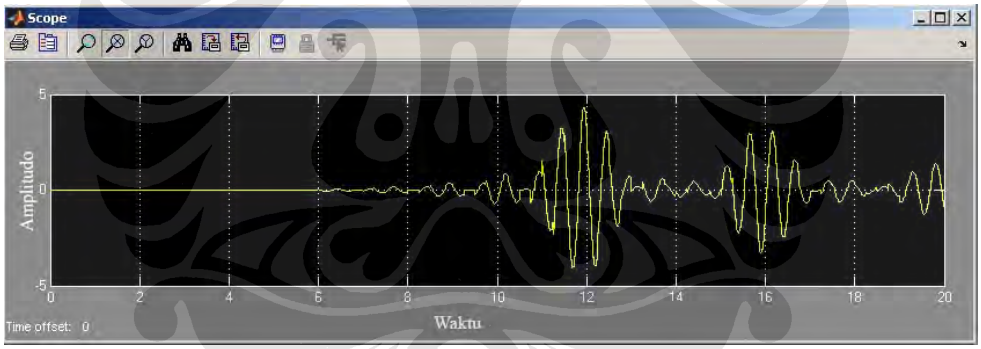
Hasil keluaran sinyal dari DSK yang diplot dengan menggunakan Microsoft Excel dapat dilihat pada Gambar 4.3.



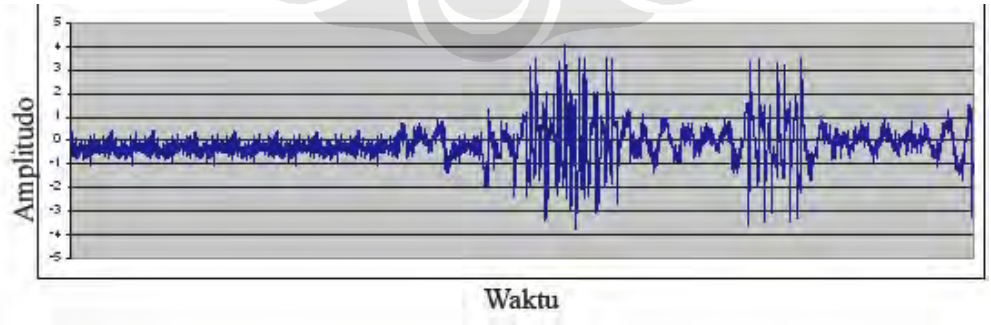
Gambar 4.3 Hasil plot keluaran dari osiloskop

4.3 PERBANDINGAN HASIL UJI COBA SIMULINK DENGAN DSK TMS320C6713

Analisis dilakukan dengan membandingkan keluaran yang dihasilkan oleh Simulink dengan keluaran yang dihasilkan oleh DSK TMS320C6713 yang ditampilkan pada osiloskop. Untuk pengambilan data, waktu uji coba di*set* pada detik ke-20 dan ke-30, baik pada Simulink maupun pada DSK TMS320C6713.

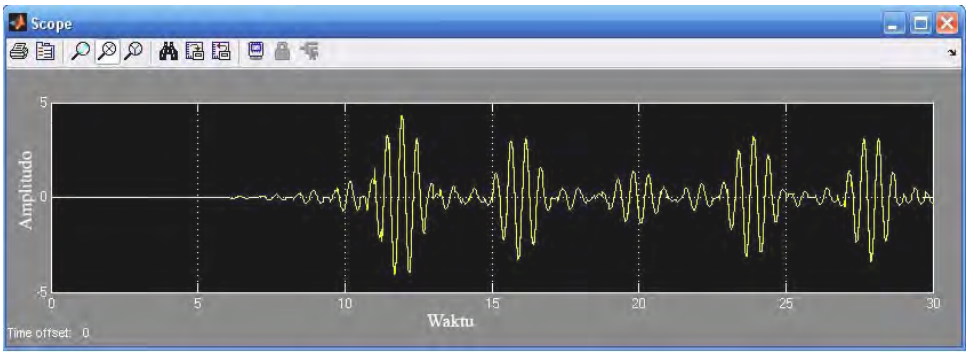


(a)

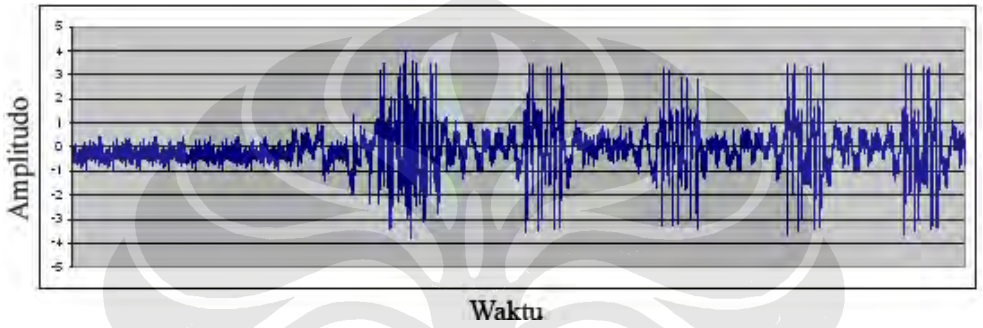


(b)

Gambar 4.4 Perbandingan hasil sinyal pada detik ke-20. (a) Hasil keluaran pada Simulink. (b) Hasil keluaran dari line out DSK TMS320C6713.



(a)



(b)

Gambar 4.5 Perbandingan hasil sinyal pada detik ke-30. (a) Hasil keluaran pada Simulink. (b) Hasil keluaran dari line out DSK TMS320C6713.

Dari kedua grafik tersebut, terlihat bahwa sinyal baru muncul setelah detik ke-8 baik pada hasil Simulink maupun keluaran DSK TMS320C6713 karena adanya *delay* pada model sistem. Dari perbandingan kedua gelombang ini, terdapat perbedaan bentuk sinyal. Hal ini terjadi karena adanya *noise* yang mengakibatkan munculnya *ripple* kecil dan lonjakan *ripple* pada gelombang yang dihasilkan. *Noise* ini disebabkan oleh *grounding* yang kurang baik pada *probe* pengukuran osiloskop. Selain itu, pengukuran yang dilakukan dengan cara menempelkan langsung *probe* pada *lineout* DSK juga dapat menyebabkan noise.

Sinyal keluaran DSK TMS320C6713 pada osiloskop secara umum telah sesuai dengan output pada Simulink, baik dari bentuk gelombang maupun *delay* yang terjadi. Dari analisis ini dapat disimpulkan bahwa rancang bangun sistem 16-QAM yang dilakukan telah sesuai dengan teori dan telah berfungsi sebagaimana mestinya.

BAB V
KESIMPULAN

Terdapat beberapa kesimpulan yang dapat diambil dari rancang bangun 16-QAM pada DSK TMS320C6713 hingga uji coba yang dilakukan, yaitu antara lain:

1. Dari analisis, dapat disimpulkan bahwa modulator 16-QAM dapat diaplikasikan pada DSK TMS320C6713.
2. Dari hasil analisis perbandingan hasil rancang bangun pada Simulink dan DSK TMS320C6713, terdapat sedikit perbedaan bentuk gelombang akibat adanya *noise*.
3. *Noise* terjadi karena *grounding* yang kurang baik pada *probe* pengukuran osiloskop.

DAFTAR ACUAN

[1] Hewlett Packard, Application Note 1298, "Digital Modulation in Communications Systems –An Introduction", printed in U.S.A,1997.

[2] Danielson, G.L., Walker, R.S.,"Transmission Principles for Technicians 2". London, Butterworth, 1981.

[3] K.Kisiel, D. Sahota, G. Swaminathan, "*Quadrature* Amplitude Modulation :A simulation study", Simon Fraser University, Canada, 2005.

[4] Wikipedia Foundation, Inc., "*Quadrature* Amplitude Modulation", http://en.wikipedia.org/wiki/QAM, 30.08.2007.

[5] Washington, Gregory and Arun Rajagopalan, "Simulink Tutorial", Ohio State University, 23 April 2003.

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

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

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

[9] Brown, D Richard, "Digital Signal Processing and Applications with the TMS320C6713 DSK", Worcester Polytechnic Institute, 2007.

[10] MATLAB R2007a, Help documentation, 2007.

DAFTAR PUSTAKA

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

Embree, Paul M., "C Algorithms for Real-Time DSP", Prentice Hall,1995

Ingle, Vinay K. and Proakis, John G., "Digital Signal Processing Using MATLAB V.4", PWS Publishing Company, 1997.

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

Langton, C., "All About Modulation", www.complextoreal.com, 2002.

MathWorks, Inc., "Communications Blockset 2", 2000.

MathWorks Inc., "Target Support Package™ TC6 User's Guide", 2008.

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

Proakis, John G., Manolakis, Dimitris G., "Introduction to Digital signal Processing", Macmillan Publishing Company, 1988.

Shanmugam, K. Sam, "Digital and Analog Communication Systems", John Willey & Sons.inc, 1985.

Spectrum Digital, Inc., "TMS320C6713 DSK Technical Reference", 2003.

Texas Intruments, "TMS320C6000 Code Composer Studio Tutorial", February, 2008

LAMPIRAN

Lampiran 1 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('ichan','w');

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
load ('sumber.mat');
uk=size(source);
ukuran=uk(1);
enable(rx,'ichan');
waktu=ukuran*0.02;
if isenabled(rx,'ichan')
rx.writemsg('ichan',uint8(source))
end
pause(100);
RTDXcleanup(cc,rx);
```

```
%=======================================================
% 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,'ichan');
disable(rx);             % disable RTDX

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