RANCANG BANGUN RANGKAIAN PENERIMA OFDM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASIS SIMULINK

SKRIPSI

OLEH

PONTAS PONCIUS SITUMORANG

0404030679

PROGRAM STUDI TEKNIK ELEKTRO

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

RANCANG BANGUN RANGKAIAN PENERIMA OFDM DENGAN MENGGUNAKAN DSK TMS320C6713 BERBASIS SIMULINK

SKRIPSI

OLEH

PONTAS PONCIUS SITUMORANG

0404030679

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

GENAP 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

RANCANG BANGUN RANGKAIAN PENERIMA OFDM
DENGAN MENGGUNAKAN
DSK TMS320C6713 BERBASIS 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, 21 Mei 2008

Pontas Poncius Situmorang
0404030679

PERSETUJUAN

Skripsi dengan judul:

RANCANG BANGUN RANGKAIAN PENERIMA OFDM
DENGAN MENGGUNAKAN
DSK TMS320C6713 BERBASIS SIMULINK

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, 21 Mei 2008
Dosen Pembimbing

Dr. Ir. Arman Djohan Diponegoro, M.Eng
NIP. 131 476 472

UCAPAN TERIMAKASIH

Segala puji dan syukur penulis panjatkan kehadirat Tuhan YME yang telah memberikan rahmat dan berkat sehingga penulis dapat menyelesaikan skripsi ini.

Penulis juga ingin mengucapkan terima kasih khususnya kepada Bapak **Dr. Ir. Arman Djohan Diponegoro, M.Eng** selaku pembimbing yang telah bersedia meluangkan waktunya untuk memberikan gagasan, konsultasi, petunjuk, saran-saran, dan motivasi serta kemudahan lainnya sehingga skripsi ini dapat diselesaikan dengan baik.

Selain itu, penulis juga mengucapkan terima kasih kepada :

1. Tuhan Yang Maha Kuasa yang telah membantu dan memberkati penulis sehingga skripsi ini bisa diselesaikan
2. Kedua orangtua, kakak, dan adik-adikku
3. Asisten laboratorium telekomukasi dan kendali serta teman-teman elektro 2004 yang selalu memberikan semangat
4. Kekasih hatiku Rahayu Setiawati Damanik, yang selalu memberikan doa, dukungan, semangat, mengingatkan dan segala sesuatu sehingga penulisan skripsi ini bisa selesai dengan baik.

Pontas Poncius Situmorang
0404030679
Depertemen Teknik Elektro

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

Rancang Bangun Rangkaian Penerima OFDM dengan Menggunakan DSDK TMS320C6716 Berbasis Simulink

ABSTRAK

Teknologi komunikasi berkembang sangat cepat saat ini. Beberapa diantaranya adalah WiMax (*Worldwide Interoperability for Microwave Access*) dan PLC (*Power Line Communication*). Kedua teknologi tersebut berkembang dengan pesat karena teknologi komunikasi yang terdapat di dalamnya, yaitu OFDM (*Orthogonal Frequency Division Multiplexing*).

Salah satu teknologi dalam bidang telekomunikasi yang memungkinkan penggunaan *bandwidth* lebih maksimal adalah OFDM. OFDM adalah salah satu teknik transmisi yang menggunakan beberapa buah *frequency subcarrier* yang saling tegak lurus (*orthogonal*). Karakteristik yang saling tegak lurus membuat *frequency* sub*carrier* dapat saling *overlap* tanpa menimbulkan interferensi dengan menggunakan teknik IFFT (*Inverse Fast Fourier Transform*).

Pada skripsi ini, dilakukan rancang bangun rangkaian penerima OFDM dengan menggunakan DSK (*Digital Signal Processing Starter Kit*) TMS320C6713 berbasis simulink. Dari hasil rancang bangun didapatkan bahwa rancang bangun rangkaian penerima OFDM dapat dibangun dengan menggunakan DSP (*Digital Signal Processing*) *Processor*.

Kata Kunci : ***Bandwidth, Orthogonal Frequency Division Multiplexing, Frequency Subcarrier, Orthogonal, DSP Processor.***

Pontas Poncius Situmorang
0404030679
Electrical Engineering Department

Supervisor
Dr. Ir. Arman Djohan Diponegoro, M.Eng

Building OFDM Receiver with DSK TMS320C6713 Based On Simulink

ABSTRACT

Technology of communication is growing fast. A few of them are WiMax (Worldwide Interoperability for Microwave Access) dan PLC (Power Line Communication). Both of them are growing fast because of the technology inside, which is OFDM (Orthogonal Frequency Division Multiplexing).

One of the technology in telecommunication that could maximize the use of bandwidth is OFDM (Orthogonal Frequency Division Multiplexing). OFDM is one of the transmission technique which use a few frequency subcarrier and also orthogonal each other. The characteristic, which is orthogonal, make frequency subcarrier could overlap without produce interference by using IFFT (Inverse Fast Fourier Transform) technique.

In this research, receiver OFDM is built using DSK (Digital Signal Processing Starter Kit) TMS 320C6713 based on simulink. The result shows that OFDM can be built by using DSP (Digital Signal Processing) Processor.

Keywords : Bandwidth, Orthogonal Frequency Division Multiplexing, Frequency Subcarrier, Orthogonal, DSP Processor.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

OFDM	Orthogonal Frequency Division Multiplexing
WiMax	Worldwide Interoperability for Microwave Access
PLC	Power Line Communication
IFFT	Inverse Fast Fourier Transform
DSK	Digital Signal Processing Starter Kit
DSP	Digital Signal Processing
FDM	Frequency Division Multiplexing
SC	Single Carrier
BPSK	Binary Phase Shift Keying
QPSK	Quaternary Phase Shift Keying
QAM	Quadrature Amplitude Modulation
DQAM	Demodulation Quadrature Amplitude Modulation
IDFT	Inverse Discrete Fourier Transform
DFT	Discrete Fourier Transform
KBytes	Kilo Bytes
KHz	Kilo Hertz
PC	Personal Computer
ALU	Arithmetic and Logical Unit
I/O	Input/Output
ADC	Analogue to Digital Converter
DAC	Digital to Analogue Converter
MIPS	Million Instructions Per Second
MFLOPS	Million Floating Point Per Second
CPLD	Complex Programmable Logic Device
EEPROM	Electrically Erasable Programmable Read Only Memory
JTAG	Joint Test Action Group
HDL	Hardware Design Language
ROM	Read-Only Memory
POST	Power On Self Test

SDRAM	Synchronous DRAM
MHz	Mega Hertz
Mcbsp	Multichannel Buffered Serial Port
LED	Light Emitting Diode
PWR	Power
CCS	Code Composer Studio
TI	Texas Instrument
RTDX	Real Time Data Exchange
USB	Universal Serial Bus
XOR	Exclusive OR
SNR	Signal to Noise Ratio

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Saat ini komunikasi menjadi kebutuhan yang sangat penting. Para pengguna jasa komunikasi mengharapkan agar komunikasi dan pengiriman data dengan jumlah yang besar dapat berlangsung dengan cepat dan berkualitas tinggi. Pengiriman data yang berukuran besar dengan cepat dibutuhkan *bandwidth* yang besar pula sedangkan *bandwidth* yang tersedia terbatas. Dibalik kekurangan tersebut, *bandwidth* yang kecil memiliki kelebihan, yaitu *power* yang dibutuhkan juga kecil. Dengan kondisi tersebut, maka berkembanglah teknologi komunikasi, khususnya teknik transmisi, yang dapat mengirim data dengan cepat dengan *bandwidth* yang kecil sehingga *power* yang dibutuhkan juga kecil. Salah satu teknik transmisi yang berkembang dan sangat bermanfaat adalah *orthogonal frequency division multiplexing* (OFDM) [1].

OFDM adalah salah satu teknik transmisi *multicarrier* yang melakukan proses pengiriman data melalui kanal transmisi yang dibagi menjadi beberapa kanal dengan dengan *frequency subcarrier* yang saling tegak lurus satu dengan yang lain sehingga dapat terjadi *overlap* antar *frequency subcarrier* tanpa menimbulkan interferensi [3]. Dengan teknik *overlap* yang terdapat pada OFDM, *bandwidth* yang terbatas tidak menjadi suatu masalah karena data dapat dikirim dengan sangat cepat dan tingkat efisiensi pemakaian *bandwidth* juga sangat tinggi.

Keunggulan tersebut yang membedakan teknik transmisi *frequency multicarrier* OFDM dengan teknik transmisi *frequency multicarrier* yang telah ada sebelumnya (konvensional). Dengan kelebihan-kelebihan yang dimiliki OFDM maka di masa depan akan semakin banyak peralatan komunikasi yang menggunakan teknologi ini.

Pada skripsi ini dilakukan proses rancang bangun OFDM dengan menggunakan *Digital Signal Processing Starter Kit* (DSK) TMS 320C6713 berbasis simulink. Rancang bangun akan menghasilkan program bahasa C yang dapat dijalankan dengan menggunakan DSK sehingga proses rancang bangun

dapat dilakukan berkali-kali tanpa biaya dibandingkan membuat sebuah chip. Hasil rancang bangun akan diuji dengan simulink dan diimplementasikan ke DSK.

1.2 TUJUAN PENULISAN

Tujuan penulisan skripsi ini adalah rancang bangun rangkaian penerima OFDM dengan menggunakan DSK TMS320C6713 berbasis simulink.

1.3 BATASAN MASALAH

Masalah dibatasi hanya pada simulasi dan rancang bangun rangkaian penerima OFDM tanpa memperhitungkan *noise* pada kanal transmisi dengan menggunakan MATLAB R2007a dan DSK TMS320C6713.

1.4 SISTEMATIKA PENULISAN

BAB I PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang penulisan, tujuan penulisan, batasan masalah, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan mengenai *Frequency Division Multiplexing* (FDM), *Orthogonal Frequency Division Multiplexing* (OFDM), simulink, dan DSK TMS320C6713.

BAB III RANCANG BANGUN

Di bab ini akan dipaparkan rancang bangun penerima OFDM dengan menggunakan simulink dan DSK TMS320C6713.

BAB IV UJICOBA DAN ANALISIS

Pada bab ini diperlihatkan hasil rancang bangun serta analisis.

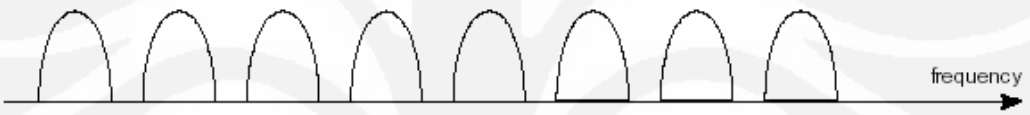
BAB V KESIMPULAN

Memberikan kesimpulan dari skripsi.

BAB II
TINJAUAN PUSTAKA

2.1 *FREQUENCY DIVISION MULTIPLEXING* (FDM)

Frequency Division Multiplexing (FDM) adalah awal mula perkembangan OFDM *(Orthogonal Frequency Division Multiplexing).* FDM adalah salah satu teknik transmisi yang proses pengiriman informasinya dilakukan dengan membagi kanal menjadi beberapa frekuensi yang berbeda. *Frequency subcarrier* yang satu dengan yang lain diberi jarak agar tidak terjadi *overlap* sehingga tidak terjadi interferensi seperti dapat dilihat pada Gambar 2.1.

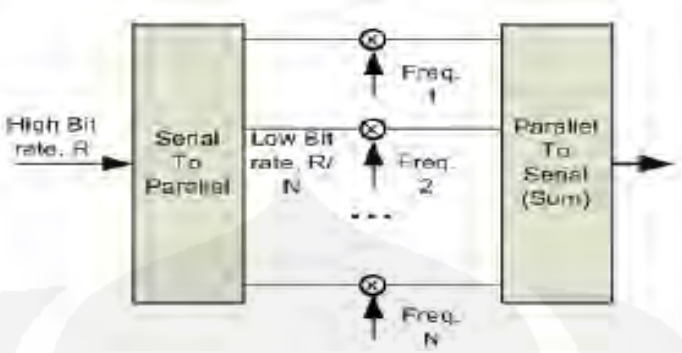


Gambar 2.1 Konvensional FDM *multicarrier technique* [1]

Pemberian jarak antar frekuensi tentu membuat pemakaian *bandwidth* menjadi tidak maksimal karena terdapat selang frekuensi yang tidak terpakai. Hal ini sangat merugikan karena efisiensi pemakaian *bandwidth* menjadi tidak maksimal.

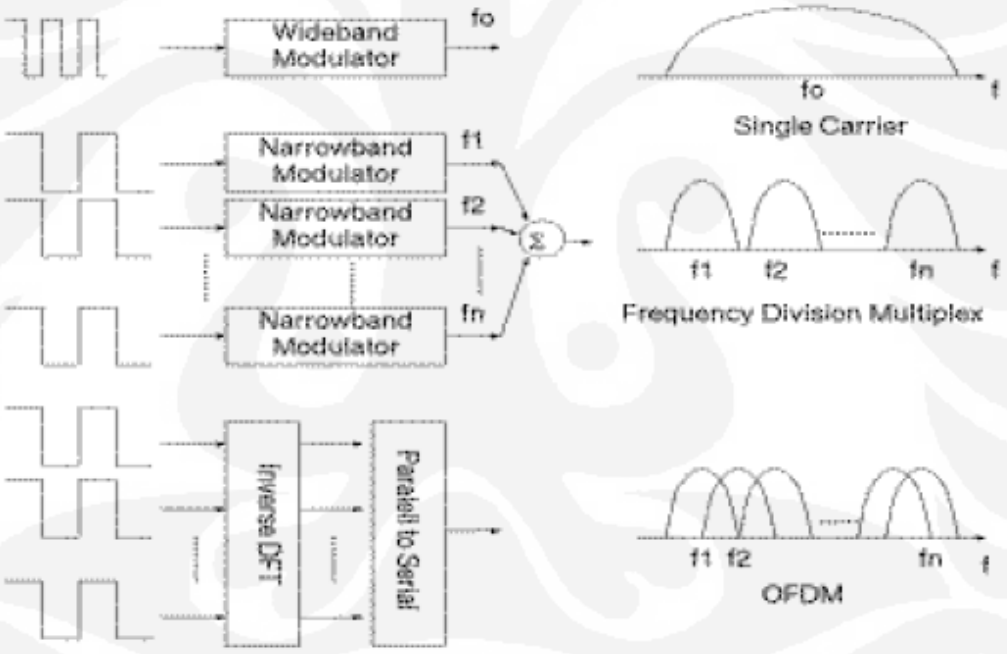
2.2 *ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING* (OFDM)

Orthogonal Frequency Diviosion Multiplexing (OFDM) adalah teknik transmisi yang berkembang dari teknik FDM. Perbedaan utama keduanya terletak dari segi efisiensi pemakaian *bandwidth*. Perbedaan tersebut disebabkan cara kerja OFDM yang lebih baik dibandingkan FDM. Pada OFDM, informasi dengan *high data rate* dibagi menjadi beberapa bagian informasi *low data rate* dan menggunakan *frequency subcarrier* yang saling tegak lurus *(orthogonal)* antara yang satu dengan yang lain seperti yang dapat dilihat pada Gambar 2.2.



Gambar 2.2 *Multicarrier* FDM [2]

Untuk memperjelas perbedaan OFDM dengan FDM dan sistem *Single Carrier (SC)* baik dari operasi dasar maupun efisiensi spektrum bisa dilihat pada Gambar 2.3.



Gambar 2.3 Perbandingan SC, FDM, dan OFDM [3]

Gambar 2.3 menunjukkan bahwa OFDM adalah salah satu jenis dari *multicarrier* yang memiliki efisensi pemakaian *bandwidth* yang jauh lebih baik. Pada OFDM, *overlap* antar *frequency subcarrier* yang bersebelahan diperbolehkan karena masing-masing saling *orthogonal* sehingga efisiensi pemakaian *bandwidth* sangat tinggi, sedangkan pada sistem *multicarrier*

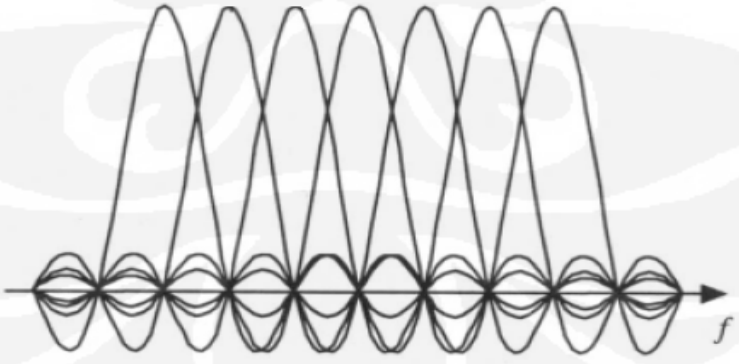
konvensional untuk mencegah interferensi antar frekuensi yang bersebelahan harus diselipkan frekuensi penghalang (*guard band*) sehingga menyebabkan menurunnya kecepatan transmisi dan pemakaian *bandwidth* menjadi tidak maksimal. Selain perbedaan tersebut, pada *multicarrier* konvensional juga diperlukan *bandpass filter* sebanyak frekuensi yang digunakan, sedangkan pada OFDM cukup menggunakan FFT saja.

Pemakaian *frequency subcarrier* yang saling tegak lurus pada OFDM memungkinkan *overlap* antar *frequency subcarrier* tanpa menimbulkan interferensi satu dengan yang lain. *Orthogonal* disini mengandung makna hubungan matematis antar *frequency subcarrier* yang dapat dinyatakan dengan persamaan matematika pada persamaan 2.1 dan 2.2.

$$\int_a^b \psi_p(t)\psi^*_q(t)dt = 0 \quad \text{untuk} \quad p \neq q \tag{2.1}$$

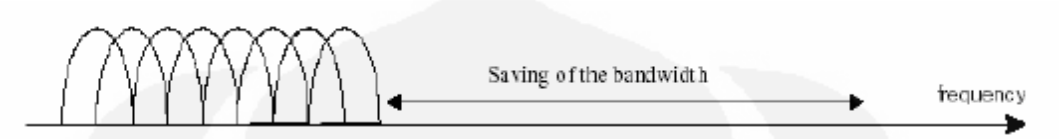
$$\int_a^b \psi_p(t)\psi^*_q(t)dt = K \quad \text{untuk} \quad p = q \tag{2.2}$$

O*rthogonal* pada OFDM memiliki arti bahwa spektrum frekuensi dari suatu *subcarrier* bernilai nol saat spektrum frekuensi *subcarrier* yang bersebelahan bernilai maksimum seperti digambarkan pada Gambar 2.4.



Gambar 2.4 Spektrum OFDM [4]

Overlap yang terdapat pada OFDM menjadikan proses pengiriman data menjadi lebih cepat dan tingkat efisiensi pemakaian *bandwidth* yang sangat tinggi seperti yang dapat dilihat pada Gambar 2.5.



Gambar 2.5 OFDM *multicarrier technique* [1]

2.3 PRINSIP KERJA OFDM

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

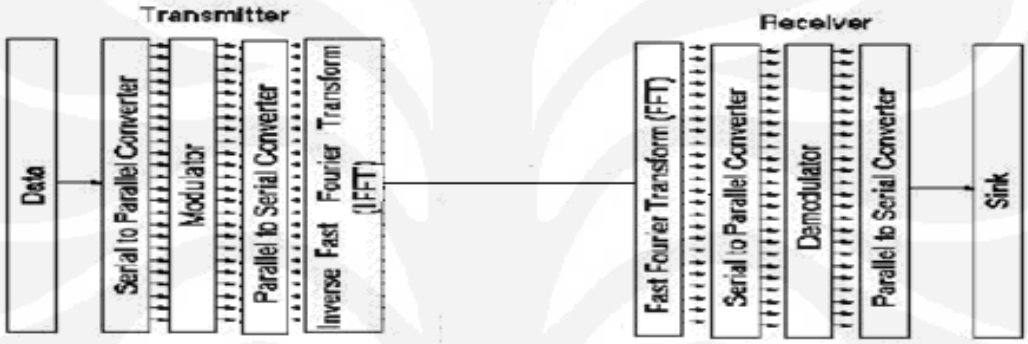
Sinyal yang telah termodulasi tersebut kemudian diaplikasikan ke *Inverse Discrete Fourier Transform* (IDFT) untuk pembuatan simbol OFDM dan mengubah *frequency domain* menjadi *time domain*. Setelah itu, simbol-simbol OFDM dikonversikan lagi kedalam bentuk serial kemudian sinyal dikirim. Sinyal yang dikirim dapat dinyatakan melalui persamaan 2.3.

$$s(t) = \mathrm{Re}\left\{ \sum_{n=-\infty}^{+\infty} b_n f(t-nT) e^{j(\omega_0 t+\varphi)} \right\} \qquad (2.3)$$

Penggunaan IDFT ini memungkinkan pengalokasian frekuensi yang saling tegak lurus (*orthogonal*) dan mengubah domain sinyal dari *frequency domain* ke *time domain* untuk selanjutnya sinyal dikirim ke *receiver* OFDM. Penerapan IDFT dapat dinyatakan melalui persamaan 2.4.

$$X(nT) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\exp(jk2\pi\frac{n}{N}) \qquad (2.4)$$

Sinyal OFDM dibangkitkan dan diproses secara digital. Hal ini dimaksudkan untuk mengurangi kerumitan menyediakan sejumlah besar osilator. Oleh karena itu, proses modulasi dan demodulasi dilakukan dengan teknik pengolahan digital, yakni *Discrete Fourier Transform* (DFT). Blok dasar OFDM yang lengkap dapat dilihat pada Gambar 2.6.



Gambar 2.6 Blok dasar OFDM [3]

2.4 *RECEIVER* OFDM

Pada sisi *receiver*, dilakukan operasi yang berkebalikan dengan apa yang dilakukan pada sisi *transmitter*. Pada sisi *receiver*, sinyal yang diterima akan diaplikasikan ke *Discrete Fourier Transform (DFT)* sehingga domain sinyal berubah menjadi domain frekuensi kembali. Penerapan DFT ini dapat dinyatakan melalui persamaan 2.4.

$$X(k) = \sum_{n=0}^{N-1} X(nT)\exp(-jk2\pi\frac{n}{N}) \qquad (2.5)$$

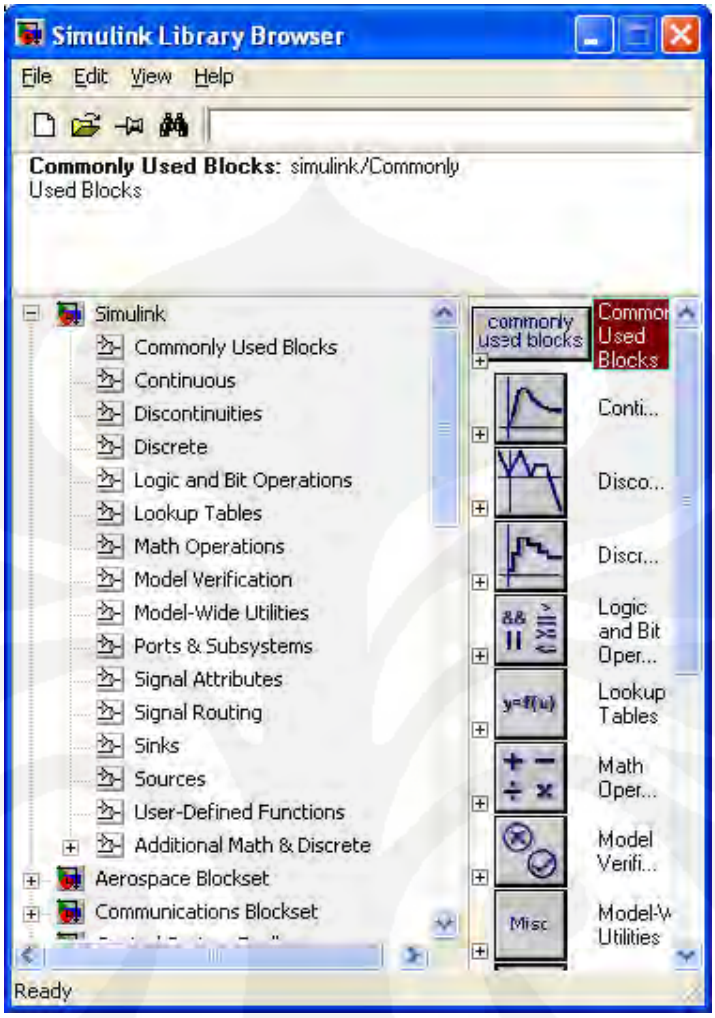
Setelah itu, sinyal dikonversikan menjadi paralel kemudian dilakukan proses demodulasi untuk mendapatkan sinyal asli informasi kembali. Sinyal yang telah dimodulasi akan memiliki persamaan 2.5.

$$s(t) = I(t)\cos(2\pi f_0 t) + Q(t)\sin(2\pi f_0 t) \qquad (2.6)$$

Untuk mendapatkan kembali sinyal informasi yang asli maka dilakukan proses demodulasi. Proses demodulasi dilakukan dengan cara melakukan operasi perkalian s(t) dengan cos untuk mendapatkan I(t) dan dengan sin untuk mendapatkan Q(t). Selanjutnya sinyal dikonversikan kembali ke serial sehingga didapatkan deretan data informasi.

2.5 SIMULINK

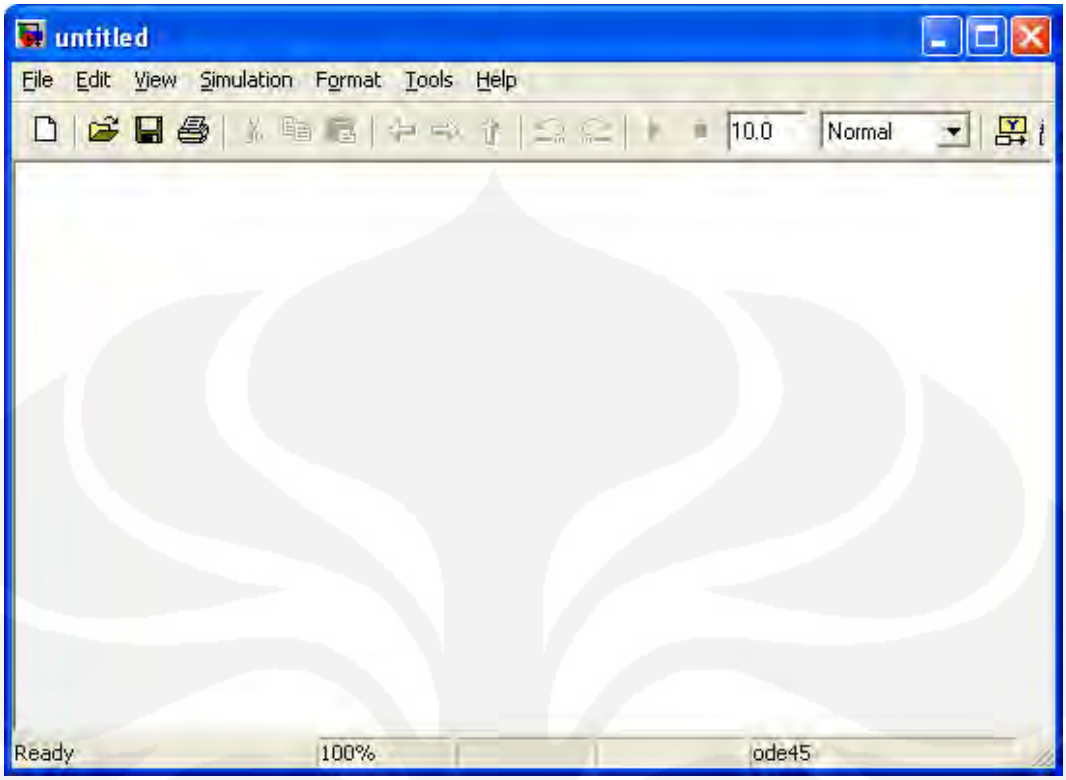
Simulink merupakan salah satu fitur yang diberikan oleh MATLAB. Simulink adalah perangkat lunak yang digunakan untuk melakukan pemodelan, simulasi, dan analisis dari sebuah sistem. Simulink dapat melakukan berbagai macam simulasi. Dengan simulink proses simulasi dapat dilakukan dengan mudah dan cepat. Simulink memiliki *library* yang terdiri dari banyak *blockset* dan *toolbox* yang dapat digunakan untuk membuat simulasi itu sendiri. Beberapa diantaranya adalah *signal processing, communication,* dan *target for TI C6000 blockset*. Pada *blockset dan toolbox* terdapat banyak blok yang memiliki fungsi-fungsi khusus seperti dapat dilihat pada Gambar 2.7.



Gambar 2.7 Simulink *library browser*

Simulink memiliki tiga *mode* proses simulasi, yakni *normal*, *accelerator*, *dan external*. *Mode normal* digunakan untuk melakukan simulasi. Mode *accelerator* digunakan untuk melakukan proses simulasi dalam waktu yang lebih cepat dibandingkan mode normal. *Mode external* digunakan untuk melakukan proses simulasi yang terhubung dengan alat di luar komputer.

Untuk mendapatkan blok-blok yang akan digunakan pada proses simulasi, kita dapat membuka simulink *library browser* yang berisi banyak *toolbox* dan blok-blok fungsi. Kita hanya butuh mengklik lalu tahan dan bawa blok ke dalam lembar kerja simulink yang dapat dilihat pada Gambar 2.8.



Gambar 2.8 Lembar kerja simulink

2.6 *DIGITAL SIGNAL PROCESSING STARTER KIT* (DSK) TMS320C6713

Digital Signal Processing Starter Kit (DSK) TMS320C6713 adalah salah satu DSP tipe C6000 yang dapat bekerja pada *fixed-point* maupun *floating-point*. Akan tetapi DSP ini masih berupa *starter kit*, yaitu suatu *platform* yang dapat mensimulasikan DSP C6713 yang sebenarnya. DSK ini lebih ditujukan untuk keperluan edukasi, penelitian, serta evaluasi. Namun, hasil dari aplikasi yang kita buat di DSK ini sangat mungkin untuk diterapkan pada DSP C6713 yang sebenarnya.

2.6.1 *Digital Signal Prcessing Starter Kit* (DSK) Prosesor

Prosesor yang digunakan oleh DSK adalah prosesor yang khusus dibuat untuk memproses sinyal secara digital. Prosesor yang digunakan sebagai DSP berbeda dengan mikroprosesor pada umumnya (*general purpose microprocessor*). DSP mempunyai sifat-sifat serta karakteristik yang unik bila dibandingkan dengan mikroprosesor biasa, seperti:

a. Operasi Matematika

Sebuah prosesor DSP dapat melakukan operasi matematika jauh lebih cepat bila dibandingkan dengan mikroprosesor biasa. Hal ini disebabkan karena DSP memiliki unit-unit aritmatika, logika, dan *discrete multiplier* yang lebih banyak. Unit-unit tersebut didesain untuk bekerja dengan kecepatan tinggi sehingga berbagai operasi matematika dapat dilakukan hanya dalam satu *cycle* dari *clock*. Sehingga DSP mampu melakukan operasi *multipl -accumulate* yang banyak digunakan dalam pemrosesan sinyal *digital*.

b. Pemakaian memori yang hemat

Pemrograman DSP tidak memerlukan memori internal yang besar. Hal ini disebabkan karena dengan kecepatannya yang tinggi, DSP dapat menggunakan memori dengan lebih efisien. Prosesor DSP yang paling modern saat ini hanya mempunyai memori sekitar 8 *KBytes* sampai 256 *KBytes*.

c. Pengolahan data kontinu

DSP dapat diaplikasikan pada proses pengolahan dimana data terus mengalir secara kontinu (*stream*).

d. Konsumsi daya rendah

Mikroprosesor biasa memerlukan daya yang relatif besar. Contohnya dapat kita lihat pada *Personal Computer* (PC). Mikroprosesor PC mengkonsumsi daya sekitar 10 Watt sampai 100 Watt. Dengan konsumsi daya sebesar ini, jika diaplikasikan pada alat yang menggunakan baterai AA, maka umur baterai mungkin hanya sampai 11 menit. Sebaliknya, prosesor DSP hanya mengkonsumsi daya sekitar 100 mW. Dengan konsumsi daya ini, baterai AA dapat dipakai sampai sekitar 18 jam.

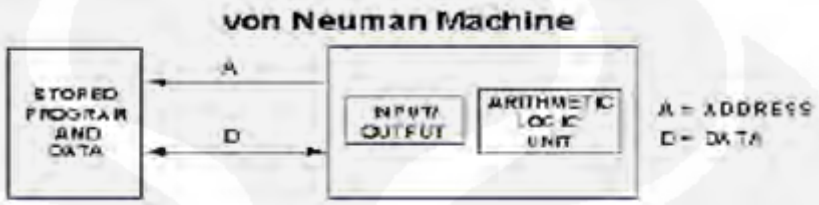
e. Real-time

DSP dapat melakukan pemrosesan sinyal secara *real time*. Hal ini disebabkan karena DSP dapat mengolah aliran data yang datang secara terus menerus. *Real time* juga dapat diartikan tidak adanya jeda waktu (*delay*) antara *input* dan *output*, sehingga saat data dimasukkan ke sebuah sistem, saat itu pula data diproses dan keluar dari sistem.

2.6.2. Arsitektur DSP Prosesor

Pada dasarnya, DSP adalah sebuah mikroprosesor juga, sehingga memiliki arsitektur tertentu. Secara umum, terdapat dua arsitektur komputer yang sudah banyak dikenal yaitu Von Neuman dan Harvard .

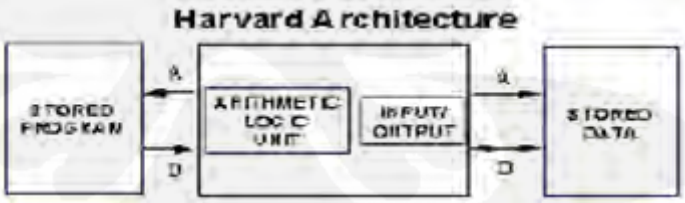
a. Von Neuman



Gambar 2.9 Arsitektur Von Neuman [4]

Pada arsitektur Von Neuman seperti pada Gambar 2.9, terdapat 2 unit operasi dasar, yaitu *Arithmatic and Logical Unit* (ALU) dan *Input/Output* (I/O). ALU berfungsi untuk melakukan operasi aritmatika dan logika, sedangkan I/O berfungsi untuk mengatur keluar masuknya data. Ciri arsitektur ini adalah program dan data disimpan pada memori yang sama.

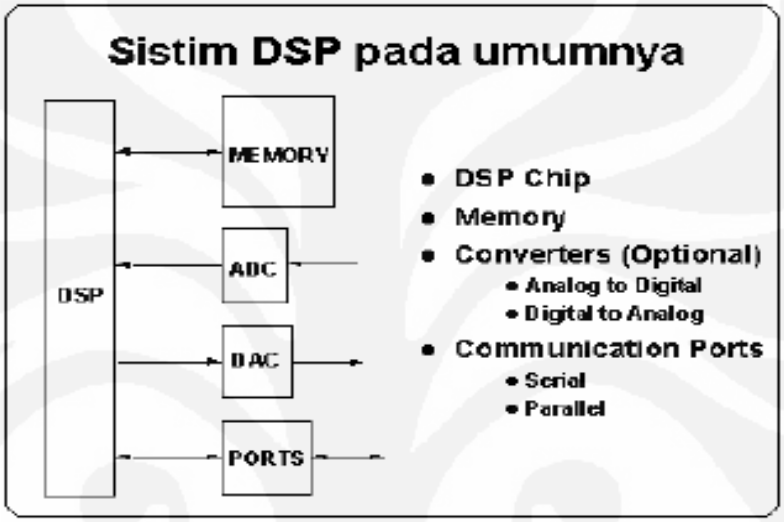
b. Harvard



Gambar 2.10 Arsitektur Harvard [4]

Unit dasar operasi yang digunakan sama dengan arsitektur sebelumnya. Perbedaannya terletak pada alokasi memorinya. Dapat dilihat pada Gambar 2.10 bahwa pada arsitektur Harvard, alokasi memori untuk program dan data berada pada lokasi yang terpisah. Oleh karena itu, pada arsitektur ini instruksi dan data dapat ditransfer secara bersamaan sehingga meningkatkan kecepatan proses.

Dari kedua arsitektur diatas, arsitektur yang banyak digunakan untuk prosesor DSP adalah arsitektur Harvard. Hal ini disebabkan karena aplikasi DSP memerlukan kecepatan proses yang tinggi agar dapat memroses sinyal secara *real time*. Namun pemisahan lokasi memori program dan data ini menyebabkan perlunya tambahan pin-pin serta memori yang digunakan. Sehingga harga prosesor DSP di pasaran menjadi lebih tinggi dari prosesor biasa.

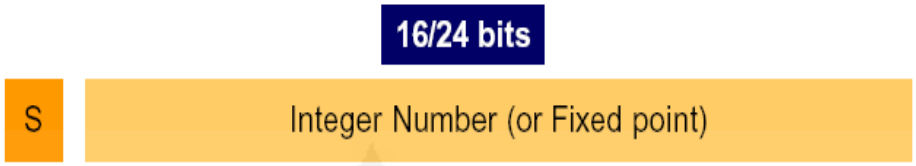


Gambar 2.11 Sistem DSP [4]

Gambar 2.11 menunjukkan sistem DSP secara umum. Sistem DSP terdiri dari memori, *Analog to Digital Converter* (ADC), *Digital to Analog Converter* (DAC), *port-port* komunikasi, serta prosesor DSP itu sendiri. Secara umum, prosesor DSP dapat dibedakan menjadi 2 tipe, yaitu *fixed point* dan *floating point*. Masing-masing memiliki arsitektur yang berbeda dan masing-masing memiliki kelebihan, tergantung dari aplikasinya.

a. *Fixed point* DSP

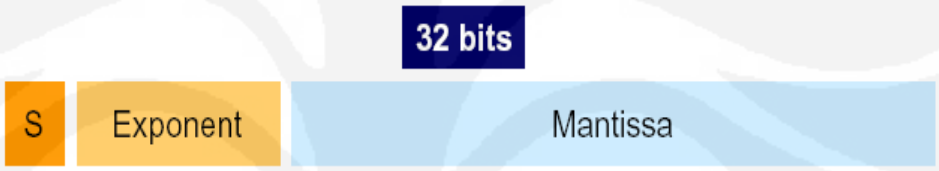
   Prosesor DSP jenis *fixed point* merepresentasikan angka dalam suatu bentuk yang presisi dengan format sepert Gambar 2.12. Misalnya, prosesor *fixed point* 16 bit memiliki jangkauan nilai $\pm 2^{15}$ dan memiliki nilai presisi, yaitu dari 1 sampai 32768.



Gambar 2.12 *Fixed point* [5]

b. *Floating point* DSP

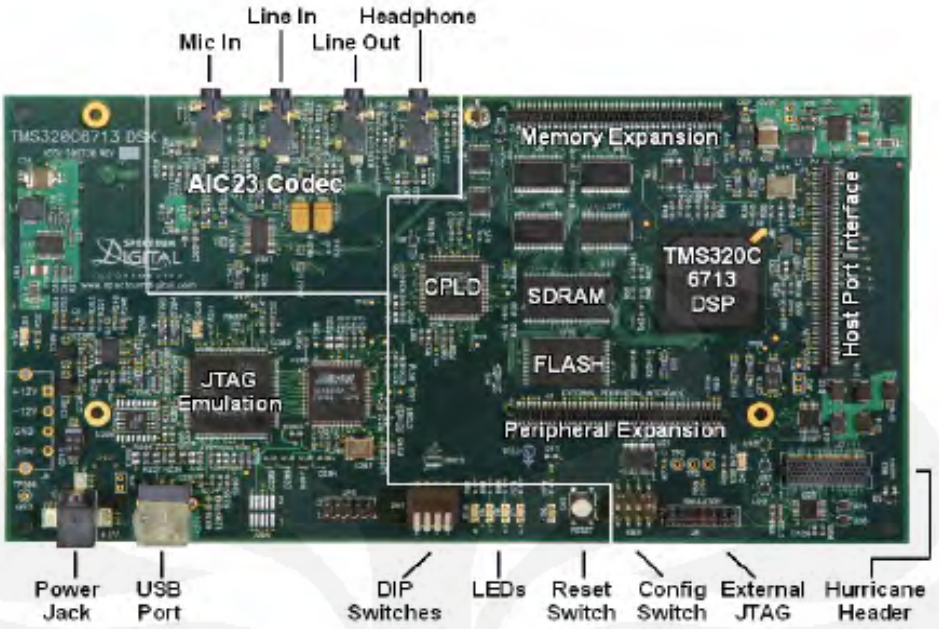
Floating point merepresentasikan angka dalam format mantisa dan eksponen seperti dapat dilihat pada Gambar 2.13. Metode ini menghasilkan jangkauan yang lebih luas. Namun, prosesor ini lebih lambat dan lebih mahal dibandingkan dengan *fixed point*.



Gambar 2.13 *Floating point* [5]

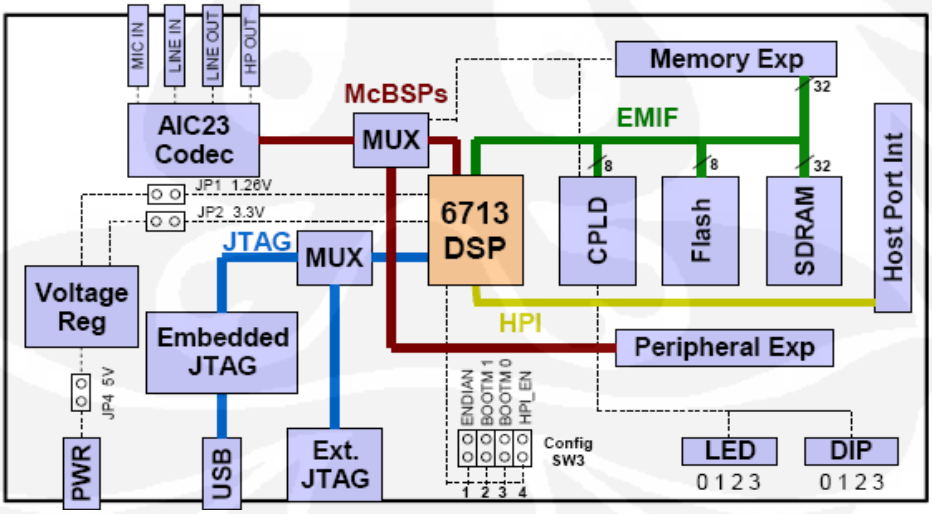
2.6.3 Komponen Utama DSK TMS320C6713

C6000 adalah tipe yang memiliki performa paling tinggi, yaitu 1200 sampai 8000 MIPS untuk *fixed point* (C64x dan C62x) dan 600 sampai 1800 MFLOPS untuk *floating point*(C67x). Tipe ini biasa digunakan untuk telekomunikasi, audio, dan video. Bentuk fisik dari DSK TMS320C6713 dapat dilihat pada Gambar 2.14.



Gambar 2.14 Bentuk fisik DSK TMS320C6713 [6]

Pada Gambar 2.14 terlihat beberapa komponen komponen pendukung dari prosesor DSP itu sendiri. Tiap komponen dihubungkan dengan jalur-jalur tersendiri yang dapat dilihat pada blok diagram pada Gambar 2.15.



Gambar 2.15 Blok diagram DSK TMS320C6713 [6]

Komponen-komponen DSK TMS320C6713 antara lain :

1. *Processor* TMS320C6713

   Merupakan *processor* dengan kecepatan *clock* 225 Hz yang mendukung operasi *fixed point* dan *floating point*. Kecepatan operasinya dapat mencapai 1350 juta operasi *floating point* per detik (MFLOPS) dan 1800 juta instruksi per detik (MIPS). Selain itu, prosesor ini dapat melakukan 450 juta operasi *multiply accumulate* per detik.

2. CPLD (*Complex Programmable Logic Device*)

   CPLD berisi *register-register* yang berfungsi untuk mengatur fitur-fitur yang ada pada *board*. DSK C6713 menggunakan Altera EPM3128TC100-10 *Complex Programmable Logic Device* (CPLD) untuk mengimpelemtasikan :

   a. Empat *Memory Mapped contro*l/status register yang dapat membuat *software* mengatur berbagai macam kelebiham/fitur dari *board* tersebut
   b. Mengatur *interface* dan sinyal yang dihasilkan oleh *daughter-cards*
   c. Mengimplementasikan "*glue logic*" yang menggabungkan semua komponen yang ada dalam *board* tersebut

   CPLD *logic* berguna untuk pengimplementasian fungsi secara spesifik kepada DSK. CPLD mengimplementasikan fungsi *logic* sederhana, yang membuat *user* tidak membutuhkan akan tambahan alat lain. Dalam CPLD terdapat register-register yang dapat digunakan *user* untuk melakukan *read*/*write* untuk melakukan konfigurasi terhadap *board*. Pada DSK C6713, terdapat 4 jenis *register* CPLD, yaitu:

   a. USER_REG *Register*
      Mengatur *switch* dan LED sesuai yang diinginkan *user*.
   b. DC_REG *Register*
      Mengawasi dan mengontrol *daughter card*.
   c. VERSION *Register*
      Indikasi yang berhubungan dengan versi *board* dan CPLD.
   d. MISC *Register*
      Untuk mengatur fungsi lainnya pada *board*.

EPM3128TC100-10 bekerja pada tegangan 3.3V (dengan tegangan toleransi 5V), terdapat didalamnya 100 pin QFP *devices* yang menyediakan 128 *macrocells*, 8 I/O *pin*, dan sebuah 10 ns *pin-to-pin delay*. *Device* tersebut berdasar pada EEPROM dan dan *programmable in-system* yang terhubung dengan *interface* JTAG. Sumber kode dari CPLD ditulis dengan menggunakan HDL (Hardware *Design Language*) standar industri.

3. *Flash memory*

   Flash memory adalah satu tipe memori yang tidak akan hilang isinya meskipun berada dalam kondisi *off*. Saat melakukan *read*, *Flash Memory* terlihat seperti *Read-Only Memory* (ROM) pada umumnya. *Flash* dapat dihapus langsung dalam blok yang besar. Saat sebuah blok telah dihapus, setiap *word* dapat diprogram ulang dengan menggunakan suatu *command* khusus.

   DSK menggunakan memori *flash* yang berfungsi untuk *booting*. Dalam *flash* ini berisi sebuah program kecil yang disebut *POST* (*PowerOn Self Test*). Program ini berjalan saat DSK pertama kali dinyalakan. Program *POST* akan memeriksa fungsi-fungsi dasar *board* seperti koneksi USB, *audio codec*, LED, *switches*, dan sebagainya.

   DSK menggunakan 512 *KBytes* external *flash* sebagai pilihan untuk *boot*. Lokasi *address flash memory* berada pada CE1 (alamat 0x90000000). *Flash* dihubungkan sebagai 256K dengan 16 bit untuk mendukung pilihan 16-bit boot pada DSK. Meskipun pada *software* memperlakukan *flash* sebagai 8 bit (dengan tidak memperdulikan 8-bit lainnya) agar sesuai dengan 8 bit *boot mode* pada 6713. Dengan konfigurasi seperti ini hanya 256 *KBytes* yang dapat digunakan langsung tanpa perubahan pada *software*.

4. *Synchronous* DRAM (SDRAM)

   Memori utama yang berfungsi sebagai tempat penyimpanan instruksi maupun data. DSK menggunakan 128 megabit *Synchronous* DRAM (SDRAM) dalam EMIF 32-bit. SDRAM di alokasikan pada awal CE0 (alamat 0x80000000). Jadi memori yang dapat digunakan adalah

sebanyak 16 *Mbytes*. SDRAM terintegrasi ini merupakan bagian dari EMIF dan harus dilakukan konfigurasi dalam *software* agar proses yang dijalankan sesuai. *Clock* EMIF berasal dari pengaturan PLL dan seharusnya berada pada 90 MHz. Ini dikarenakan *clock* internal PLL membutuhkan 450 MHz untuk memperoleh *clock* sebesar 225 MHz dengan dua pembagi. Sementara EMIF dengan *clock* 90MHz dengan pembagi lima.

Saat menggunakan SDRAM pengatur harus diatur untuk melakukan *refresh* satu baris dari *array* memori setiap 15.6 mikrosekon untuk mempertahankan integritas dari data. Dengan *clock* EMIF sebesar 90 MHz periodenya menjadi 1400 *bus cycles*.

5. AIC23 *Codec*

Codec mempunyai fungsi utama sebagai ADC (*Analog to Digital Converter*) dan DAC (*Digital to Analog Converter*). *Codec* pada DSK TMS320C6713 ini disebut dengan AIC23 *codec*. *Codec* melakukan proses *sampling* pada sinyal analog yang masuk dari *microphone* atau *line in* dan mengubahnya menjadi bentuk *digital* untuk diproses oleh DSP. Setelah data *digital* diproses, *codec* akan mengubah data itu menjadi analog kembali dan dikeluarkan melalui *line out* atau *headphone*.

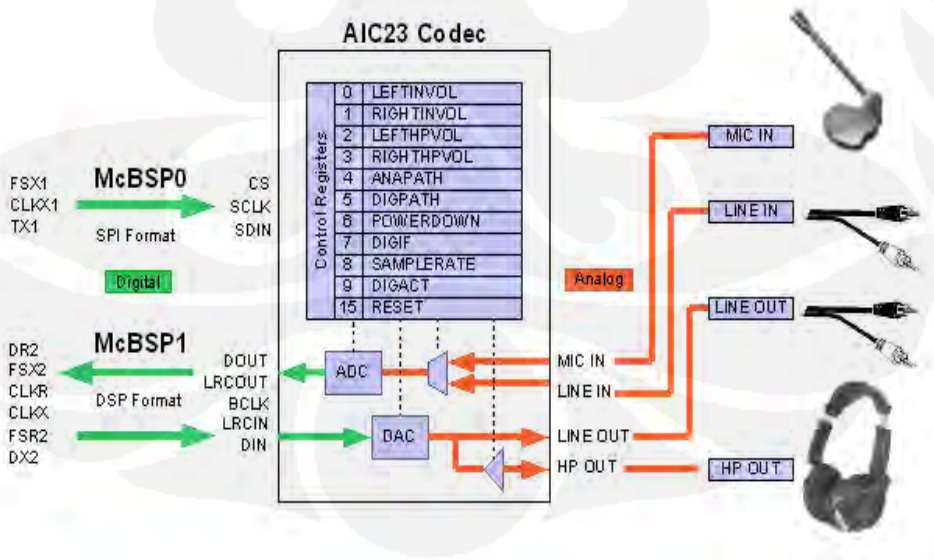
AIC23 *codec* mempunyai sepuluh register yang dapat mengontrol berbagai fungsi *codec* seperti volume, format data, *sample rate*, dan sebagainya. Selain itu terdapat juga satu register *reset*. Tiap register mempunyai ukuran sebesar 16 bit. Untuk pemakaiannya, data 16 bit ini dibagi menjadi dua bagian, tujuh bit paling kiri (*Most Significant Bit*) digunakan sebagai alamat dari register itu sendiri, sedangkan sembilan bit sisanya digunakan sebagai data mode operasi register yang bersangkutan.

Codec melakukan koneksi dengan dua serial channel. Satu untuk mengontrol konfigurasi register internal dan yang lain untuk mengirim dan menerima audio *sample digital*. McBSP0 sebagai serial *channel* pertama, berguna sebagai *control channel* yang bergerak satu arah. McBSP0 diprogram untuk melakukan pengiriman 16 bit *control word* ke AIC23 dalam format SPI. Tujuh angka pertama dalam *control word* tersebut

merupakan penentu sepesifikasi register untuk dimodifikasi, sedangkan sembilan angka terakhir menentukan nilai dari register tersebut. McBSP0 hanya digunakan pada saat konfigurasi *codec*. Pada saat pengiriman audio terjadi, McBSP0 tidak melakukan kerja apa-apa.

McBSP1 berguna sebagai data *channel* dua arah. Semua data audio yang masuk maupun yang keluar dari DSP board melalui *channel* McBSP1 Data format yang dapat didukung berdasarkan tiga variabel dari panjang *sample*, sumber sinyal *clock*, dan format data serial. Pada umumnya, contoh pada DSK menggunakan sample yang memiliki panjang *word* 16 bit dengan *codec* yang sudah dalam kondisi yang sama dengan spesifikasinya, sehingga DSP tidak perlu membantu proses apa-apa.

Codec memiliki *clock* sebesar 12 MHz. *Clock* ini merespon terhadap mode rata-rata *sample* pada USB karena pada umumnya USB memiliki besar *clock* yang sama dan dapat digunakan untuk *codec* dan USB *controller*. Rata-rata *sampling* membagi 12 MHz *clock* untuk membangkitkan frekuensi-frekuensi yang lain, seperti 96 KHz, 48 KHz, 44.1 KHz, dan 8 KHz. Rata-rata pencuplikan diatur oleh *register SAMPLERATE* pada *codec*. Gambar 2.16 menunjukkan skema dari AIC23 *codec* yang digunakan oleh DSK.



Gambar 2.16 Bagan rangkaian *codec* DSK [7]

6. *Daughter Ccard Interface*

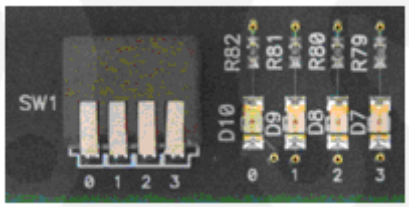
   Konektor-konektor tambahan yang berguna untuk mengembangkan aplikasi-aplikasi pada *board*. Terdapat 3 konektor, yaitu *memory expansion, peripheral expansion,* dan *Host Port Interface*.

7. *Switch* DIP

   Empat buah *Switch* DIP memperbolehkan *user* untuk memberikan pengaruh sederhana. *Switch* DIP dapat di-*read* melalui register CPLD USER_REG. *Switch* DIP ini juga dapat di-*read* menggunakan modul *Switch* DIP dari *Board Support Library*.

8. *Light Emitting Diode* (LED) *user*

   LED *user* sebanyak empat buah dapat dikendalikan oleh *user* untuk menampilkan status suatu informasi atau *feedback.* Pengaturan dilakukan dengan melakukan *write* pada *register* CPLD USER_REG. LED tersebut juga dapat diatur atau dikosongkan melalui LED *Module* pada *Board Support Library*. Gambar 2.17 menunjukkan LED *user.*



Gambar 2.17 LED *user* [6]

9. *Light Emitting Diode* (LED) Indikator Status

   LED sebagai status indikator memiliki fungsi memberitahukan status pada komponen-komponen tertentu. PWR_LED dihubungkan pada +5V *Power Supply* dan akan menyala saat *power* disambungkan. RESET_LED akan menyala bila terjadi *Reset* pada DSK. USB_IN_USE LED akan menyala bila USB terhubung pada DSK, dan akan mati bila ada external emulator diaplikasikan. USB BUSY LED mengindikasikan adanya pertukaran data atau proses terjadi pada DSK.

10. *Joint Test Action Group* (JTAG)

Joint Test Action Group (JTAG) merupakan konektor yang dapat melakukan *transfer* data dengan kecepatan yang sangat tinggi. Hal ini akan berguna dalam aplikasi *real-time*.

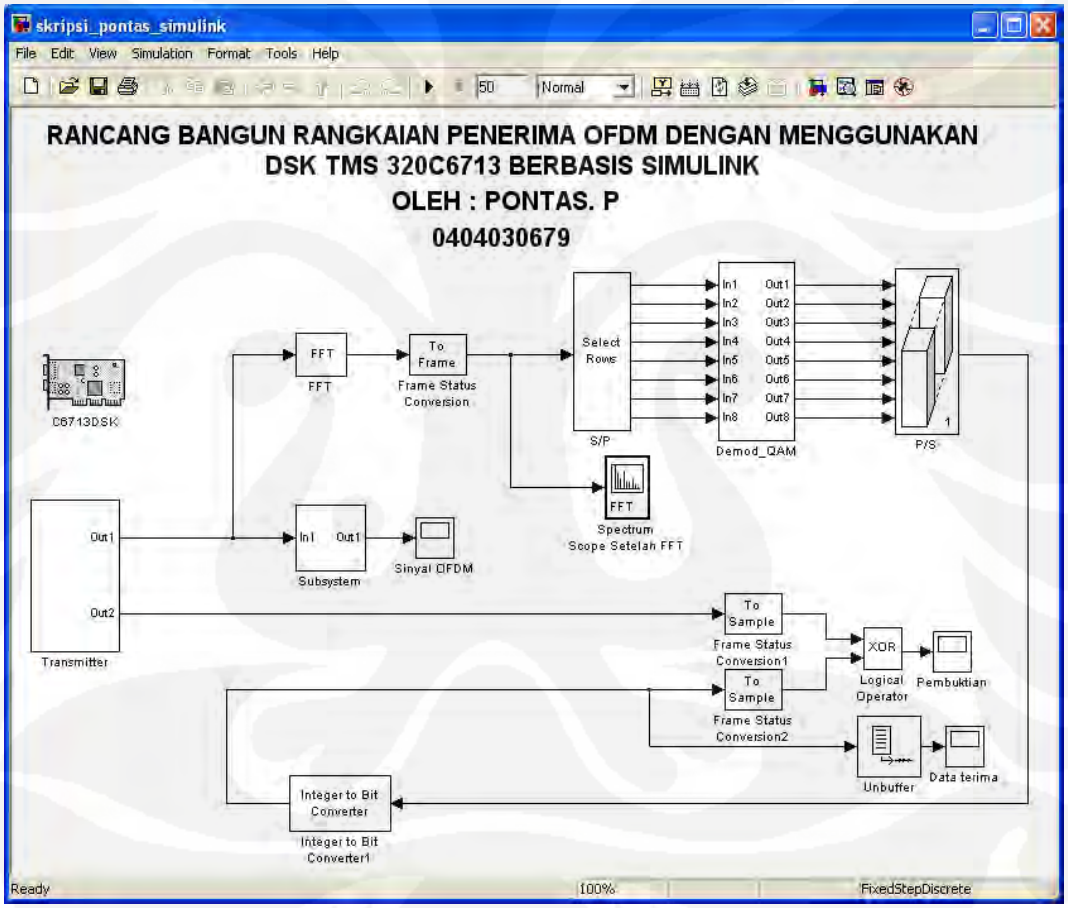
2.6.4. *Starting* DSK TMS320C6713

Pada saat DSK *board* dinyalakan, DSP prosesor akan mulai untuk melakukan *loading code* berdasarkan *boot mode* yang sudah ditentukan dari pabrik. *Power On Self Test* (POST) yang telah diprogram dalam *flash* dan *jumper* akan melakukan *boot* dari *flash memory*. POST akan terus melakukan *looping* sampai *Code Composer Studio* (CCS) diaktifkan. *Embedded* USB *controller* akan mulai aktif dan menunggu koneksi dari *software* CCS . Saat *software* DSK 6713 dibuka, secara otomatis *software* akan melakukan koneksi ke USB *port* dan bila koneksi sudah tercipta, *software* DSK 6713 CCS akan melakukan *loading* dan kemudian dapat digunakan oleh *user*. Saat melakukan koneksi dari *software* ke USB pada DSK *board*, akan muncul *Windows Hardware Device Manager*. Kemudian pada saat CCS mulai, *driver emulation* akan melakukan kontak ke DSK *board*. Lalu CCS akan melakukan proses *download emulation firmware* dari DSK. Setelah selesai, *firmware* akan memutuskan koneksi USB dan melakukan koneksi ulang sebagai emulator. Setelah kita keluar dari program CCS dan mengulang kembali proses menyalakan program, program akan otomatis melakukan koneksi melalui USB *host* ke DSK sebagai emulator dan bila DSK berada dalam kondisi menyala, program baru dapat diluncurkan. USB *controller* tidak dipengaruhi oleh tombol *reset* pada *board* DSK. Saat DSK menyala, USB *controller* mendapatkan *power* saat *reset*.

BAB III
RANCANG BANGUN

3.1 SIMULINK

Rancang bangun model rangkaian penerima OFDM dibuat dengan menggunakan simulink yang terdapat pada matlab. Model rangkaian penerima OFDM yang telah dibuat dapat dilihat pada Gambar 3.1.



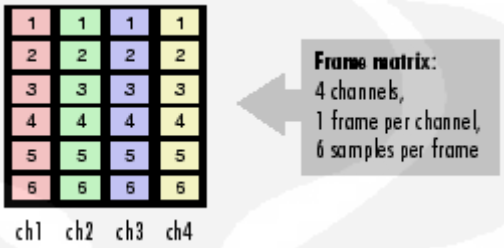
Gambar 3.1 Model rangkaian penerima OFDM rancang bangun simulink

Blok-blok yang digunakan adalah sebagai berikut :

1. *Bernoulli Binary Generator*

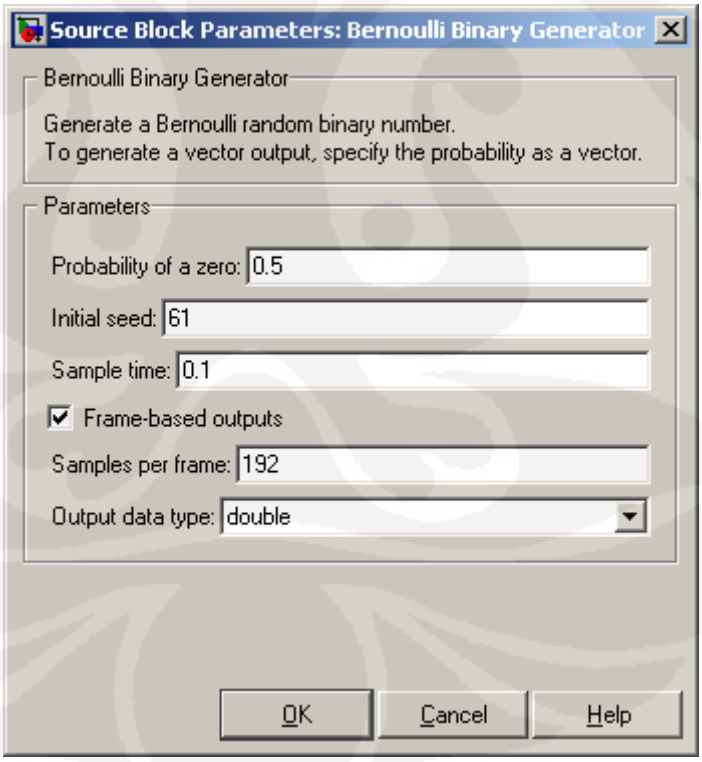
   Bernoulli Binary Generator berfungsi menghasilkan *binary bit* dengan probabilitas 0 dan 1 masing-masing adalah 0.5. Pada skripsi ini,

Bernoulli Binary Generator dibuat untuk menghasilkan bit dengan format *frame based* data. *Frame based* data memiliki arti bahwa dalam satu *frame* terdiri dari beberapa sample. Bentuk *frame based* data dapat dilihat pada Gambar 3.2



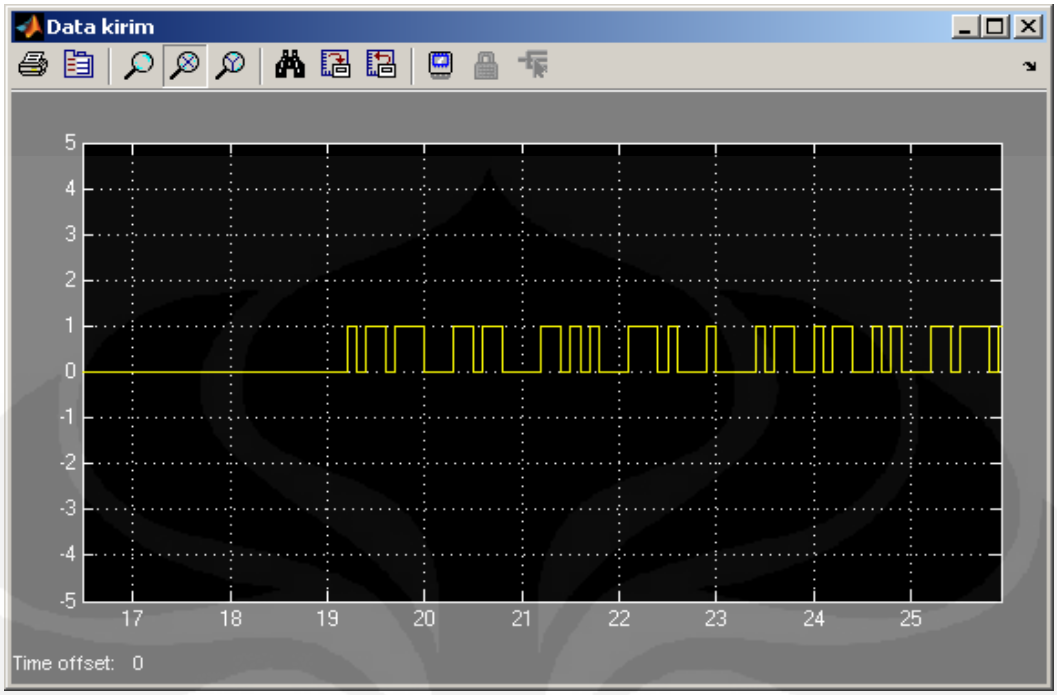
Gambar 3.2 *Frame based* data [8]

Konfigurasi parameter *Bernoulli Binary Generator* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Konfigurasi parameter B*ernoulli Binary Generator*

Dengan konfigurasi parameter seperti Gambar 3.2 didapatkan *binary bit* seperti Gambar 3.4.



Gambar 3.4 Data *input*

2. *Bit to Integer Converter*

   Bit to Integer Converter berfungsi mengubah nilai *binary* setiap beberapa bit tertentu sesuai dengan keinginan menjadi satu nilai *integer*. Pada skripsi ini, setiap enam *binary bit* diubah menjadi satu nilai *integer*. Bit-bit data yang dihasilkan oleh *Bernoulli Binary Generator* dalam satu *frame* sebanyak 192 sehingga bila enam buah *binary bit* diubah menjadi satu nilai integer akan didapatkan 32 nilai integer.

3. *Integer to Bit Converter*

   Integer to Bit Converter berfungsi mengubah setiap satu nilai *integer* menjadi nilai *binary* dengan jumlah beberapa bit sesuai dengan keinginan. Pada skripsi ini setiap satu nilai integer akan diubah menjadi enam buah *binary bit* sehingga dari 32 nilai integer akan didapatkan kembali 192 *binary bit*.

4. QAM

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

5. DQAM

   Blok DQAM berfungsi untuk melakukan proses demodulasi sinyal untuk mendapatkan sinyal asli yang dikirim dari *transmitter.*

6. IFFT

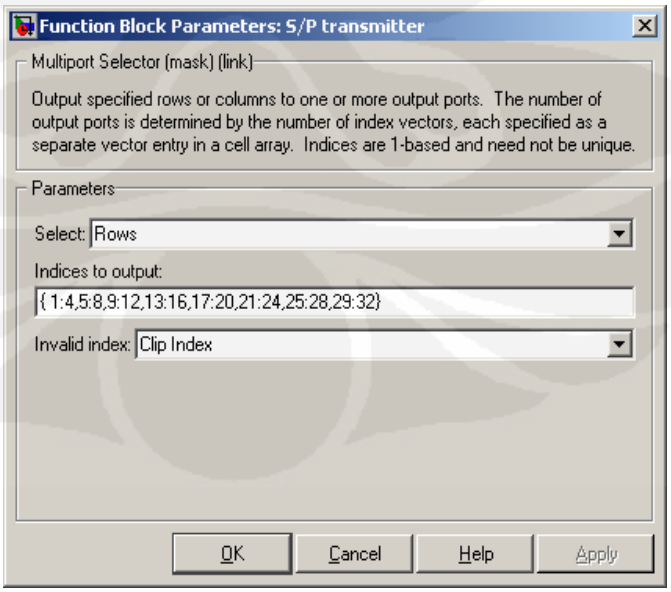
   Blok IFFT berfungsi melakukan transformasi dari domain frekuensi ke waktu.

7. FFT

   Blok FFT berfungsi melakukan transformasi dari domain waktu ke frekuensi.

8. *Multiport Selector*

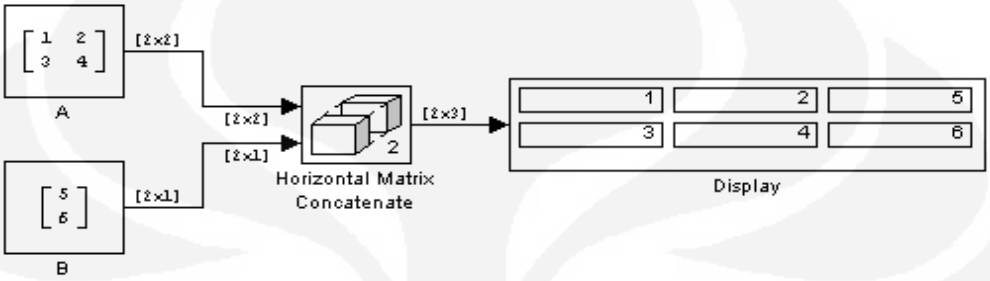
   Dalam blok *Multiport Selector* mode yang digunakan adalah *Select rows. Select rows* digunakan untuk mengubah data serial menjadi paralel. *Select rows* diisi dengan nomor urut data yang ingin diambil sesuai dengan keinginann untuk digunakan pada proses selanjutnya. Data serial berupa nilai akan diubah menjadi data paralel dengan menggunakan blok ini. Data serial berupa nilai integer dalam satu *frame* akan diubah menjadi delapan data paralel dengan menggunakan blok *select rows* sehingga masing-masing *bus* menerima empat buah nilai *integer*. Konfigurasi parameter *multiport selector* ditunjukkan oleh Gambar 3.5.



Gambar 3.5 Konfigurasi parameter *Multiport Selector*

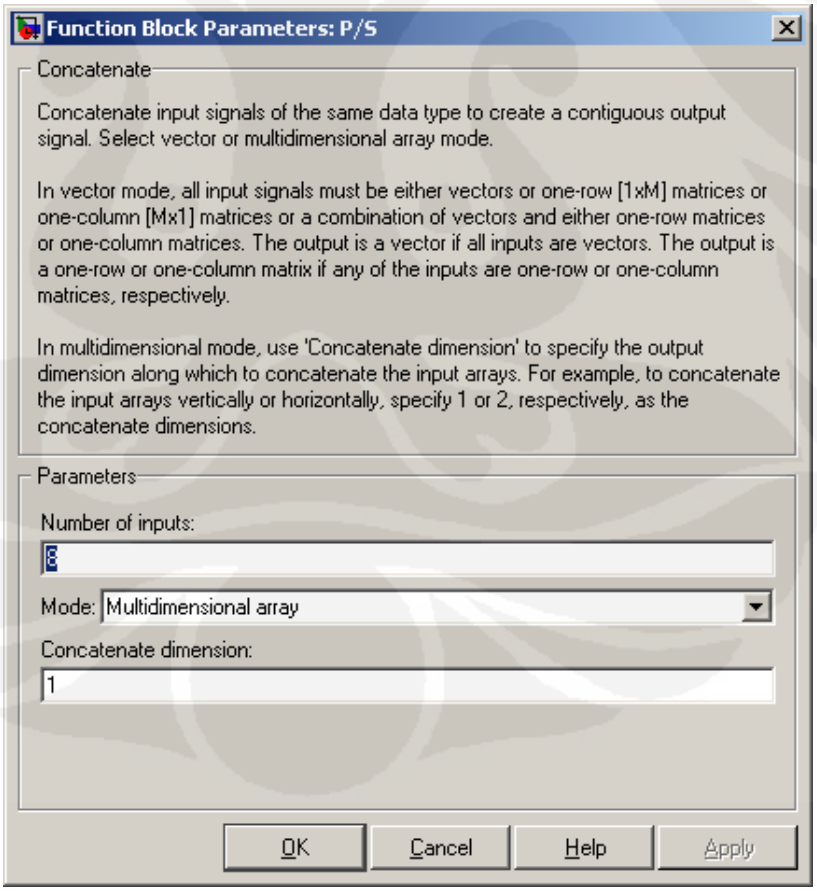
9. *Matrix Concatenation*

Matrrix Concatenation digunakan untuk mengubah data paralel menjadi data serial. *Matrix Concatenation* yang digunakan dibuat menjadi *mode multidimensional array* dengan 8 *input* dan 1 *output*. Prinsip kerja *Matrix Concatenation* dapat dilihat pada Gambar 3.6



Gambar 3.6 *Horizontal matrix concatenate* [8]

Konfigurasi parameter blok *concatenate* ditunjukkan oleh Gambar 3.7



Gambar 3.7 Konfigurasi parameter *matrix concatenate*

10. *To Workspace*

    To workspace berfungsi menyimpan hasil simulasi ke dalam *workspace*.

11. *Outport*

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

12. *To Frame*

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

13. *To Sample*

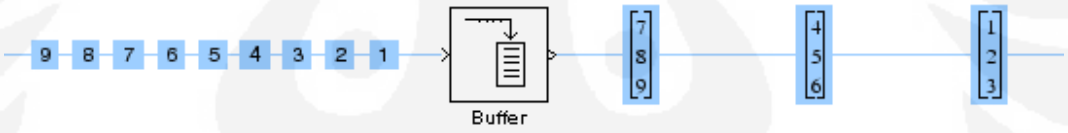
    To sample digunakan untuk mengubah *frame based* menjadi *sample based.*

14. *Spectrum Scope*

    Spectrum scope digunakan untuk melihat spektrum dari suatu sinyal.

15. *Buffer*

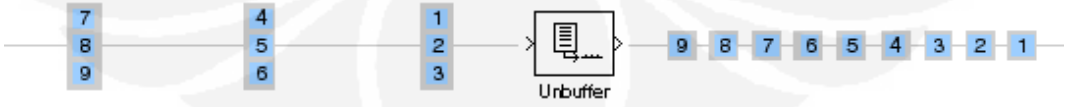
    Buffer digunakan untuk mengubah *sample based* menjadi *frame based* dengan *data rate* yang bisa lebih tinggi atau rendah sesuai dengan keinginan. Perubahan *sample based* menjadi *frame based* yang dilakukan *buffer* dapat dilihat pada Gambar 3.8.



Gambar 3.8 *Buffer* [8]

16. *Unbuffer*

    Unbuffer digunakan untuk mengubah *frame based* menjadi *sample based* dengan *data rate* yang lebih tinggi seperti pada Gambar 3.9..



Gambar 3.9 *Unbuffer* [8]

17. *Complex to Real Imaginary*

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

18. *Sine Wave*

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

19. *Rate Transition*

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

20. *Product*

    Product digunakan untuk mengalikan dua buah sinyal.

21. *Add*

    Add digunakan untuk menjumlahkan dua buah *input.*

22. *Time Scope*

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

23. *Logical Operator*

    Logical Operator digunakan untuk melakukan operasi *Boolean*. Pada skripsi ini yang digunakan operasi XOR. Data yang dibandingkan adalah data *input* dari sisi *transmitter* dengan data *output* yang telah diproses dari sisi *receiver*. Apabila nilai data *input* sama dengan data *output* maka hasil operasi XOR akan bernilai nol.

24. *Subsystem*

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

25. DAC DSK 6713

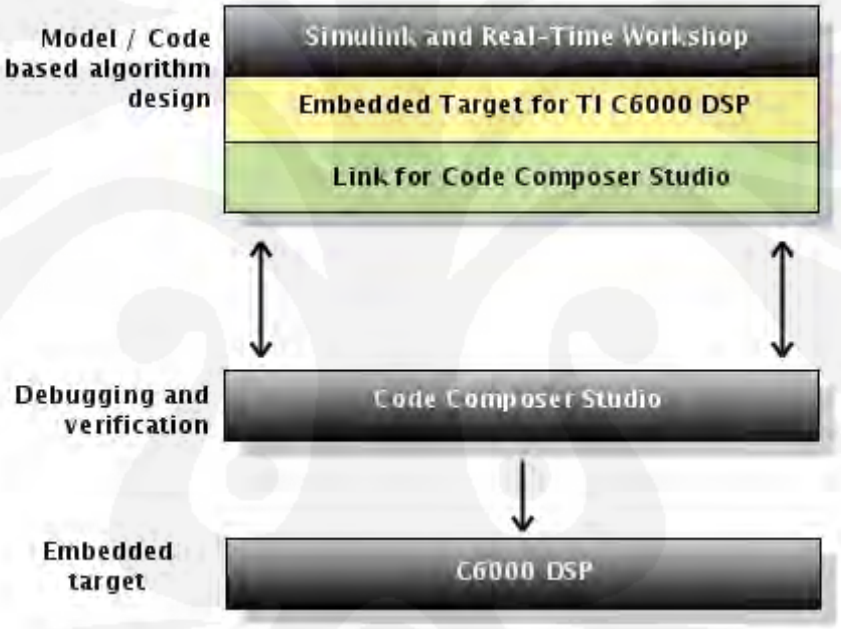
    DAC DSK 6713 dipakai agar dapat menggunakan DAC dari DSK 6713.

26. C6713 *Board*

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

3.2 DSK TMS320C6713

Rancang bangun rangkaian penerima OFDM dengan menggunakan DSK TMS320C6713 dilakukan dengan tetap berbasiskan simulink. Proses rancang bangun ini menggunakan *target preference* C6713DSK yang tersedia dalam library simulink. Selain target preference dibutuhkan *Real Time Workshop, Embedded Target for TI* (*Texas Instrument*) *C6000 DSP*, dan *Link for CCS* untuk menghubungkan simulink dengan DSK. Ketiga hal tersebut dapat ditemukan di simulink dan harus dilakukan pengaturan konfigurasi. Hubungan ketiga hal tersebut dapat dilihat pada Gambar 3.10.

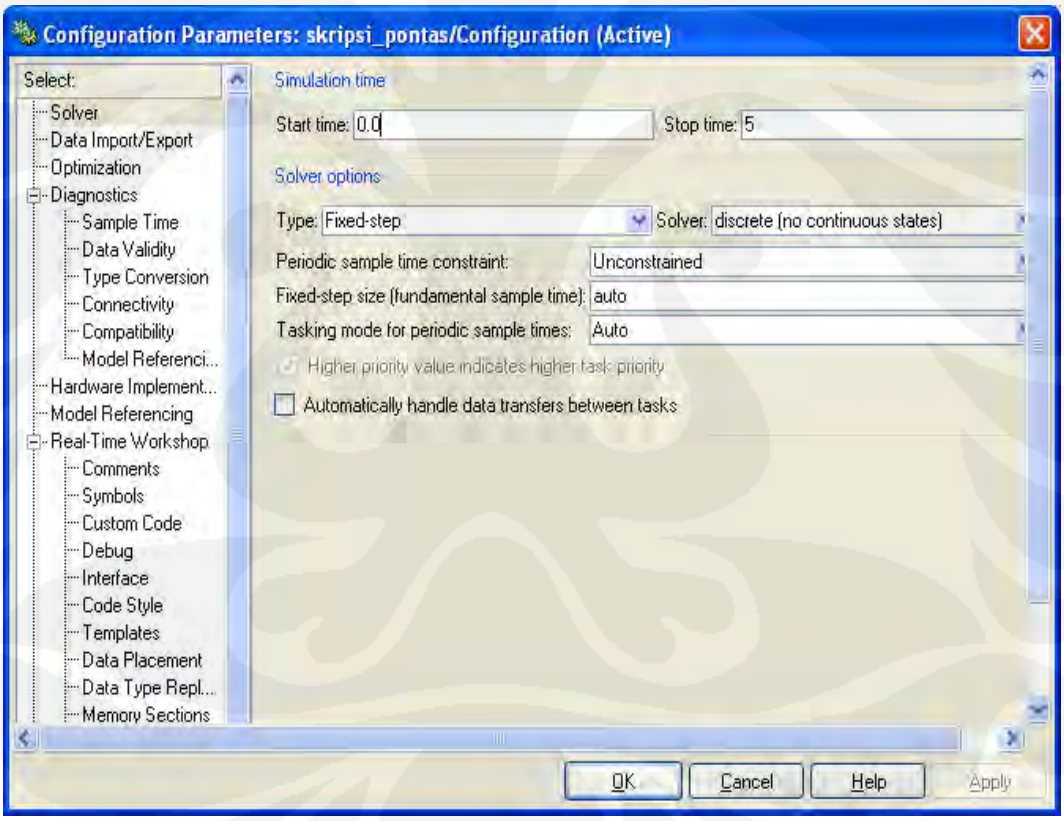


Gambar 3.10 Rancang bangun DSK [9]

Pengaturan konfigurasi parameter model rangkaian penerima OFDM terhadap ketiga hal tersebut dapat dilakukan dengan memilih menu *simulation* yang ada pada *toolbar* lalu memilih konfigurasi parameter. Selanjutnya akan muncul kotak konfigurasi parameter model seperti yang dapat dilihat pada Gambar 3.11. Pengaturan yang harus dilakukan terhadap konfigurasi parameter model rangkaian penerima OFDM yaitu :

1. Pada *tab solve*r, mengubah *type* menjadi *fixed step* dan *solver* menjadi *discrete* pada menu *solver options*.

2. Pada *tab optimization*, melakukan *uncheck block reduction* dan *implement logic signals as boolean data* yang terdapat pada menu *simulation and code geberation.*
3. Pada *tab hardware implementation*, mengubah *device type* yang terdapat pada menu *embedded hardware* menjadi TI C6000.
4. Pada *tab real time workshop* di menu *target selection*, mengubah *system target file* menjadi ccslink_ert.tlc.
5. Pada *tab real time workshop subtab debug*, melakukan *check verbose build* pada menu *build process*.
6. Pada *tab real time workshop subtab Link for CCS* di menu *project option*, mengubah *system stack size* menjadi 8192.



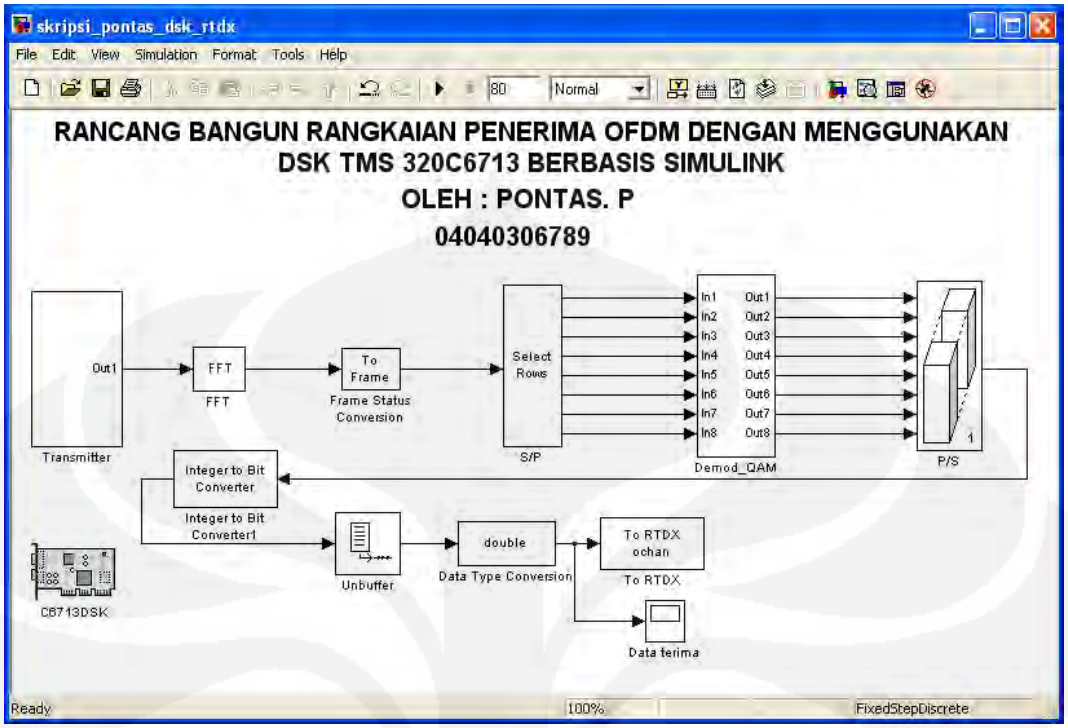
Gambar 3.11 Konfigurasi parameter model

Setelah pengaturan konfigurasi parameter *Real Time Workshop, Embedded Target for TI C6000 DSP,* dan *Link for CCS* selesai dilakukan maka hal selanjutnya yang harus dilakukan adalah melakukan beberapa perubahan

terhadap model rangkaian penerima OFDM. Perubahan tersebut yaitu menghilangkan semua *scope*, mengganti *output* yang ingin dilihat menjadi DAC DSK 6713, dan melakukan *targetting preference* model rangkaian penerima OFDM ke DSK dengan menambahkan blok C6713DSK *board* ke dalam model. Model rangkaian penerima OFDM bila *output* ingin dilihat dengan menggunakan *storage osciloscope* dapat dilihat pada Gambar 3.12.

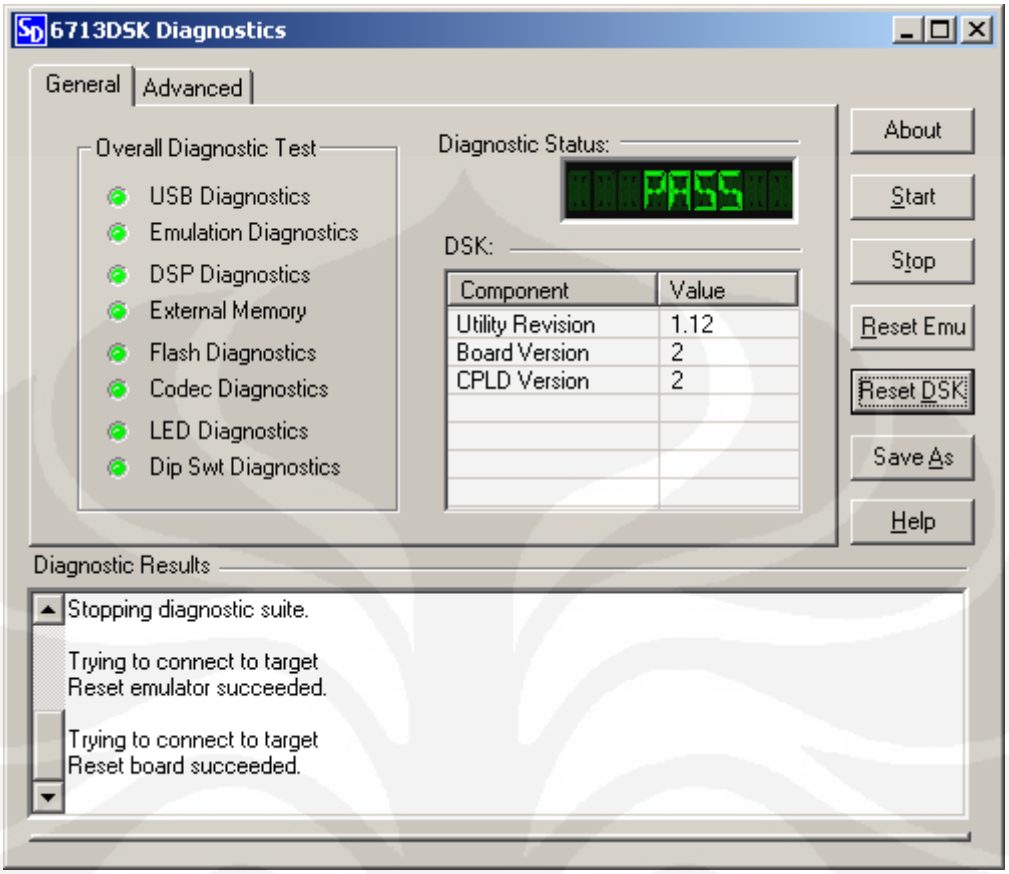
Gambar 3.12 Model rangkaian penerima OFDM dengan *storage osciloscope*

Untuk melihat *output* dengan menggunakan media komputer maka pengujian model rangkaian penerima OFDM dilakukan dengan menggunakan *Real Time Data Exchange* (RTDX) seperti dapat dilihat pada Gambar 3.13.



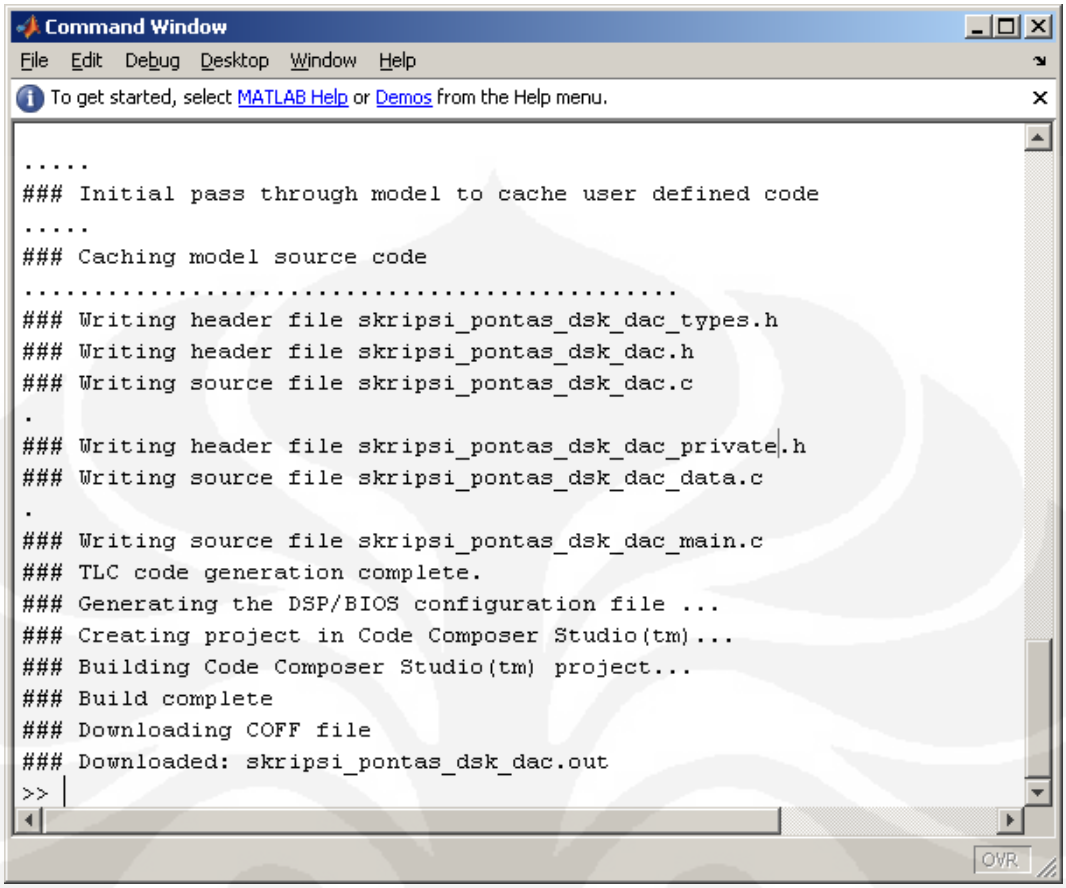
Gambar 3.13 Model rangkaian penerima OFDM dengan RTDX

Setelah model rangkaian penerima OFDM disesuaikan maka *build model* siap dilakukan. Sebelum dilakukan proses build model, komputer harus dihubungkan ke DSK dengan menggunakan kabel USB *(universal serial bus).* Untuk memastikan komputer benar-benar terhubung dengan baik ke DSK dan DSK berada dalam kondisi terbaik maka proses *diagnostic* yang terdiri dari *overall diagnostic*, *reset Emu*, dan *reset* DSK harus dilakukan. Proses *diagnostic* sangat penting dilakukan untuk mempersiapkan DSK dalam kondisi baik. Proses tersebut dapat dengan mudah dilakukan dengan menggunakan program 6713 *diagnostic* yang disediakan oleh CCS. Jika komputer terhubung dengan baik ke DSK maka pesan *PASS* akan muncul seperti dapat dilihat pada Gambar 3.14.



Gambar 3.14 *Diagnostic* proses

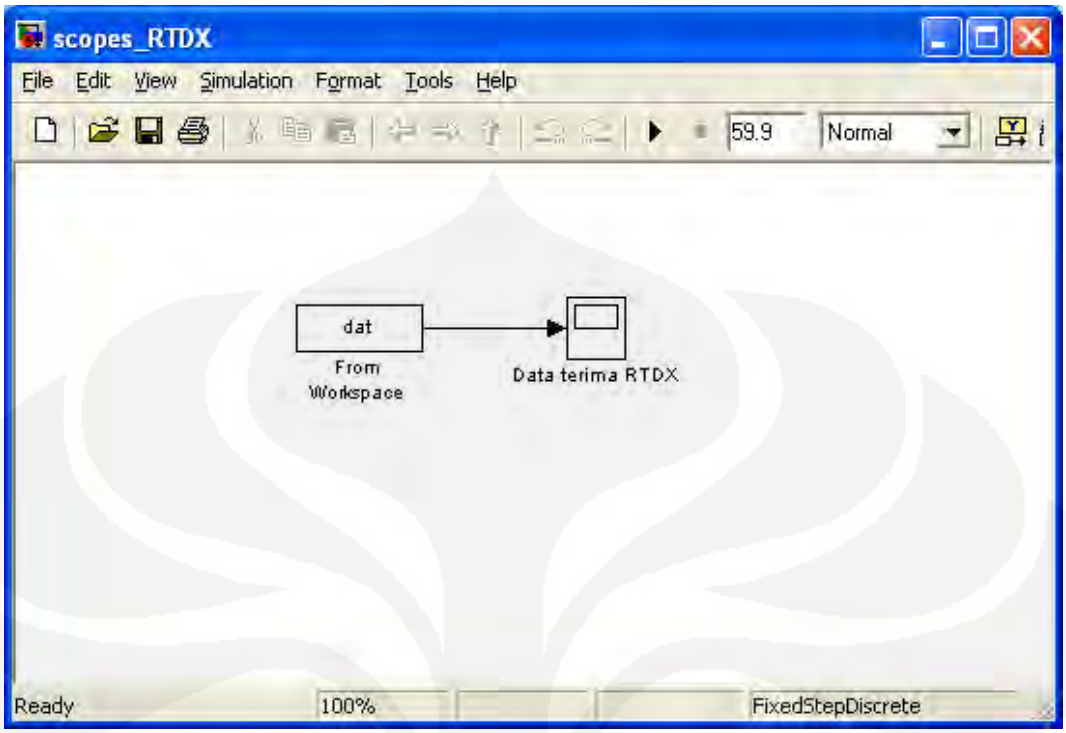
Selanjutnya dilakukan proses *Incremental Building*. *Incremental Building* dapat dilakukan dengan memilih menu yang terdapat di *tools* lalu *real time workshop* kemudian pilih *build model*. Setelah dilakukan proses *build model*, maka matlab akan membuat kode program dalam bahasa C sebagai pengganti blok simulink untuk dijalankan di DSK. Proses pembuatan program dalam bahasa C oleh matlab ditampilkan pada command window seperti terlihat pada Gambar 3.15.



Gambar 3.15 Proses pembuatan program C

Setelah program C berhasil dibuat maka program tersebut akan dibebankan ke dalam DSK.

Data *output* pada pengujian dengan menggunakan RTDX akan disimpan dengan ekstensi MAT. Data *output* tersebut dapat ditampilkan dengan menggunakan *scope* dengan menggunakan sebuah model yang dapat mengambil *input* dari *workspace* Matlab. Sebelum model dijalankan, data *output* tersebut harus berada di *workspace* Matlab. Model untuk menampilkan data *output* pada pengujian RTDX dapat dilihat pada Gambar 3.16.



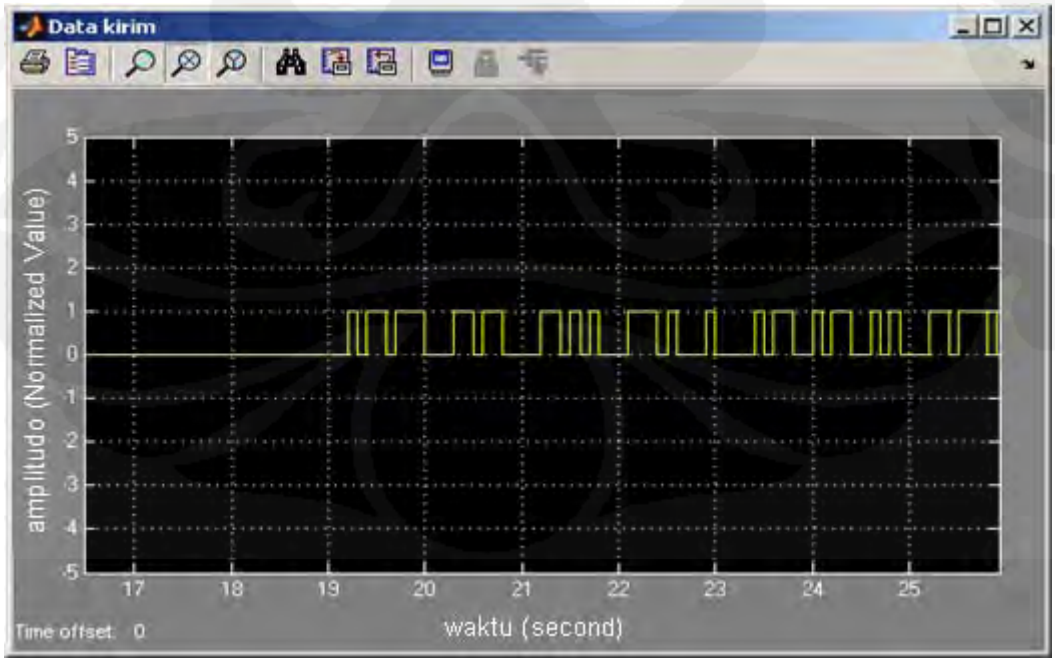
Gambar 3.16 Model pemanggil data o*utput* RTDX

BAB IV
UJI COBA DAN ANALISIS

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

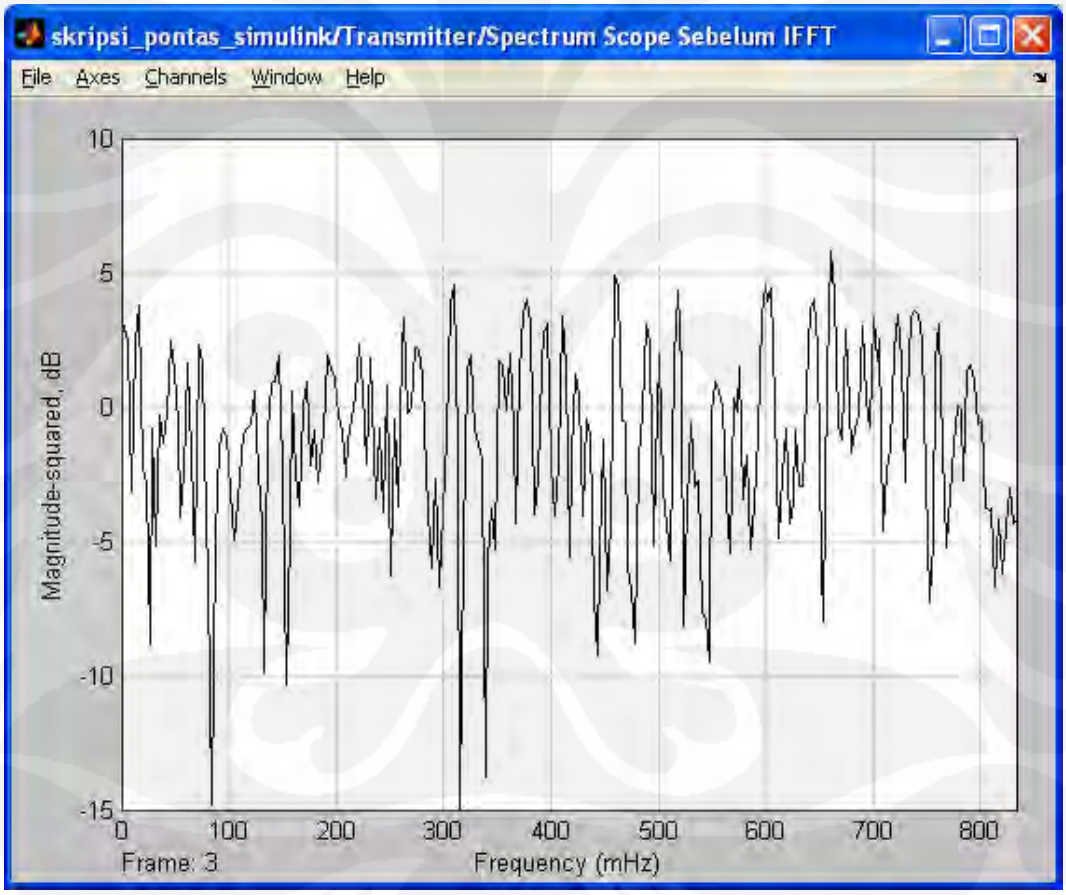
4.1 SIMULINK

Pada ujicoba simulink ini, data diambil dari sisi *receiver* untuk dibandingkan dengan data yang didapat dari sisi *transmitter*. Pada sisi *receiver*, rangkaian terdiri dari FFT, *serial to paralel converter, demodulator,* dan *paralel to serial converter*. Sinyal pada sisi *receiver* berasal dari rangkaian *transmitter* OFDM. Pada sisi *transmitter* data dibangkitkan dengan menggunakan *bernoulli binary generator* sehingga data yang dikirim hanya bernilai satu dan nol secara bergantian seperti pada Gambar 4.1.



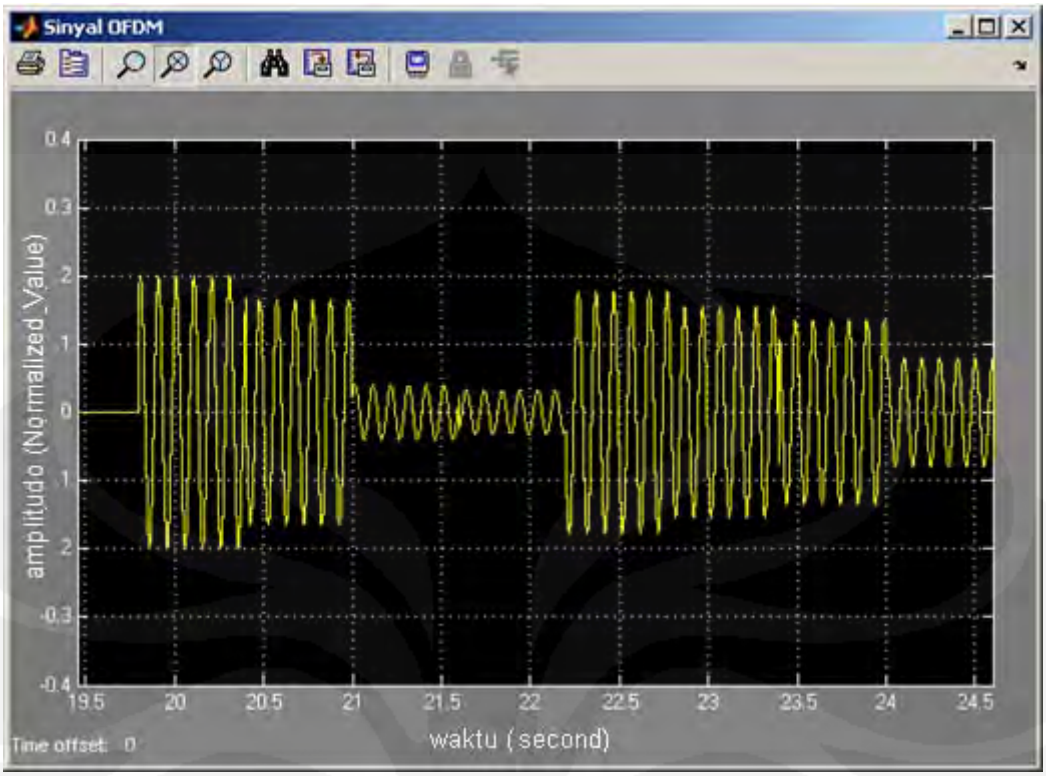
Gambar 4.1 Data kirim *transmitter*

Pada Gambar 4.1 data berupa bit-bit baru keluar setelah detik ke 19,2 karena data yang dihasilkan adalah *frame based*. Data *frame based* akan dikirim apabila jumlah *samples per frame* telah memenuhi konfigurasi yang telah dibuat sehingga untuk mendapatkan jumlah bit sesuai dengan *samples per frame* yang diinginkan, maka bit-bit data harus disimpan terlebih dahulu sampai jumlah *samples per frame* terpenuhi sehingga terjadi *delay* seperti ditunjukkan Gambar 4.1. Selanjutnya data yang dibangkitkan oleh *bernoulli binary generator* menuju blok modulasi QAM untuk dilakukan proses modulasi. Spektrum sinyal setelah dilakukan proses modulasi tampak seperti pada Gambar 4.2.



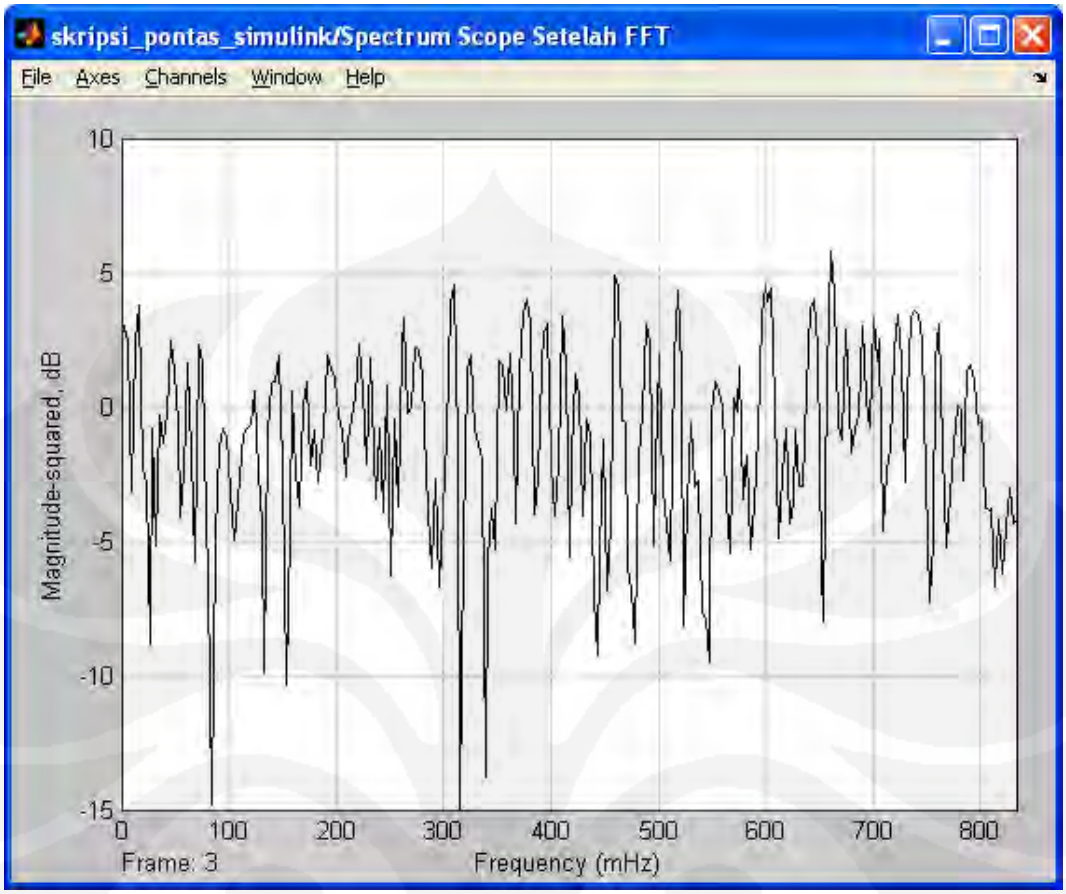
Gambar 4.2 Spektrum sinyal sebelum IFFT

Setelah dilakukan proses modulasi, kemudian sinyal menuju blok IFFT untuk proses pembentukan sinyal OFDM dan transformasi dari domain frekuensi ke domain waktu sehingga bentuk gelombang OFDM setelah melewati IFFT akan terlihat seperti Gambar 4.3.



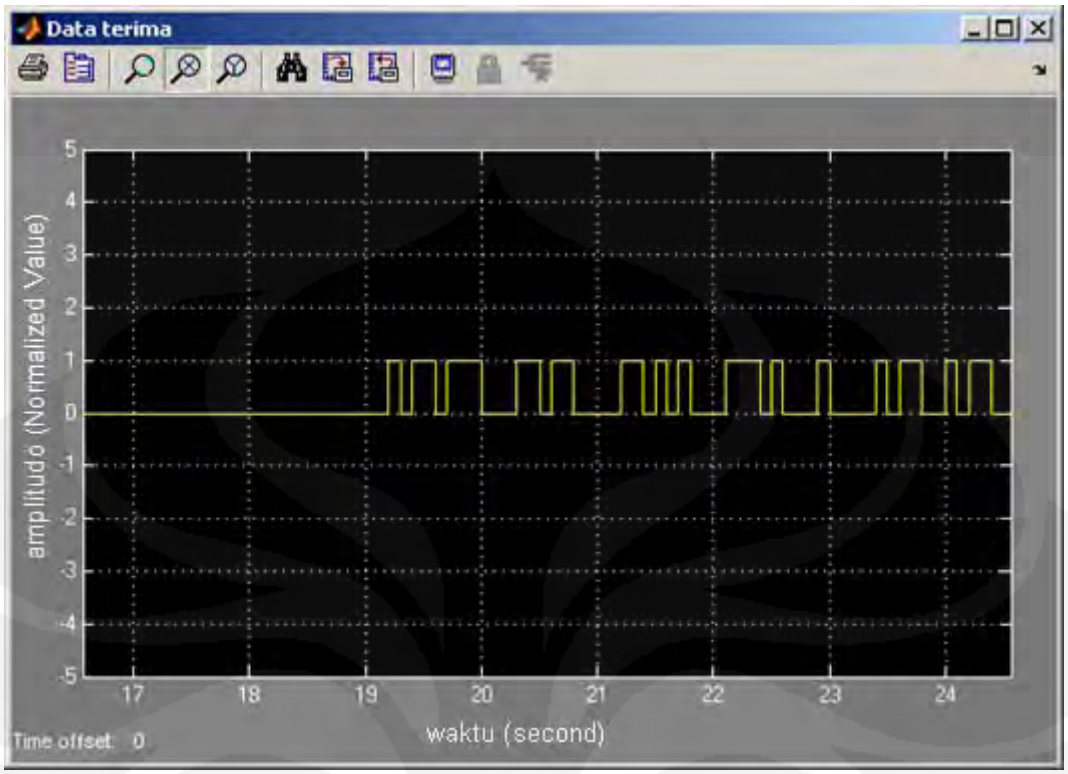
Gambar 4.3 Sinyal OFDM

Sinyal keluaran IFFT merupakan sinyal yang dikirim oleh *transmitter* OFDM. Sinyal tersebut diterima oleh *receiver* OFDM yang selanjutnya akan mengalami proses FFT dan demodulasi. Spektrum sinyal setelah melalui proses FFT dapat dilihat pada Gambar 4.4.



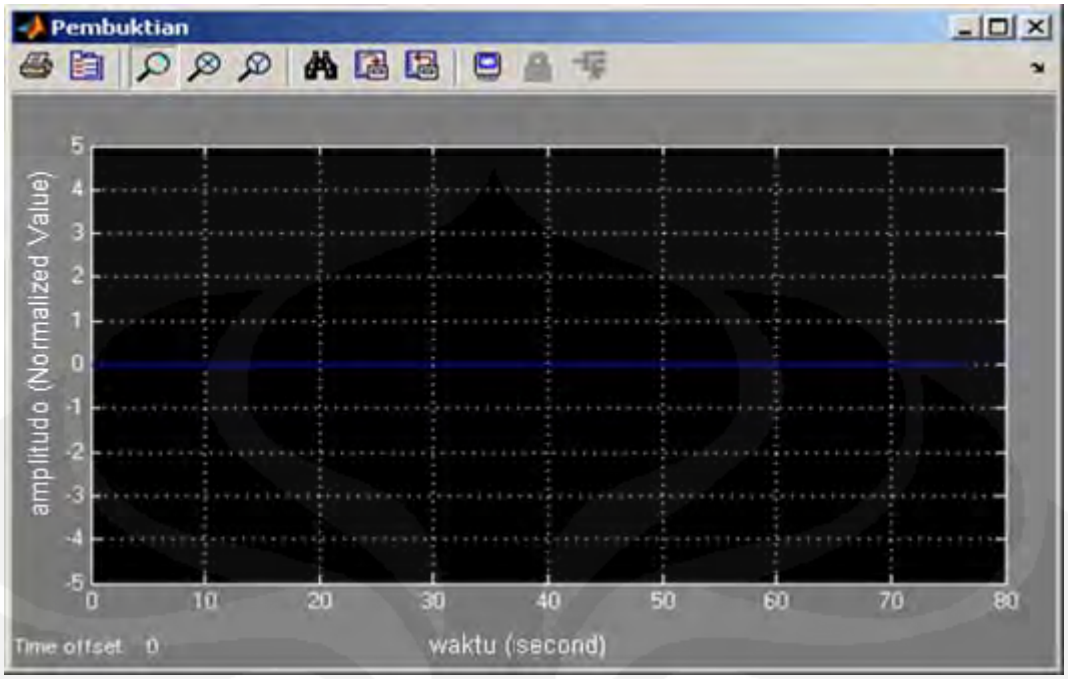
Gambar 4.4 Spektrum sinyal setelah FFT

Spektrum sinyal setelah proses FFT tidak memiliki perbedaan dengan spektrum sinyal sebelum IFFT. Hal ini berarti sinyal yang ditangkap oleh *receiver* tidak mengalami perubahan atau kerusakan. Keadaan ini bisa dicapai karena tidak adanya gangguan terhadap sinyal yang dikirim oleh *transmitter*. Setelah dilakukan proses FFT, maka selanjutnya dilakukan proses demodulasi QAM terhadap sinyal untuk mendapatkan kembali sinyal informasi yang asli. Setelah melewati proses demodulasi maka didapatkan sinyal informasi yang asli. Data hasil demodulasi dapat dilihat pada Gambar 4.5.



Gambar 4.5 Data terima *receiver*

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



Gambar 4.6 Perbandingan data

Grafik pada scope pembuktian selalu bernilai nol karena dilakukan proses XOR terhadap data yang dikirim dengan data yang diterima. Apabila data yang dikirim dengan data yang diterima bernilai sama maka hasil XOR sama dengan nol. Oleh karena data yang dikirim dengan data yang diterima selalu bernilai nol maka hasil XOR juga akan selalu bernilai nol. Hal ini menunjukkan bahwa rancang bangun model rangkaian penerima OFDM yang telah diuji sudah berfungsi dengan semestinya.

4.2. DSK TMS320C6713

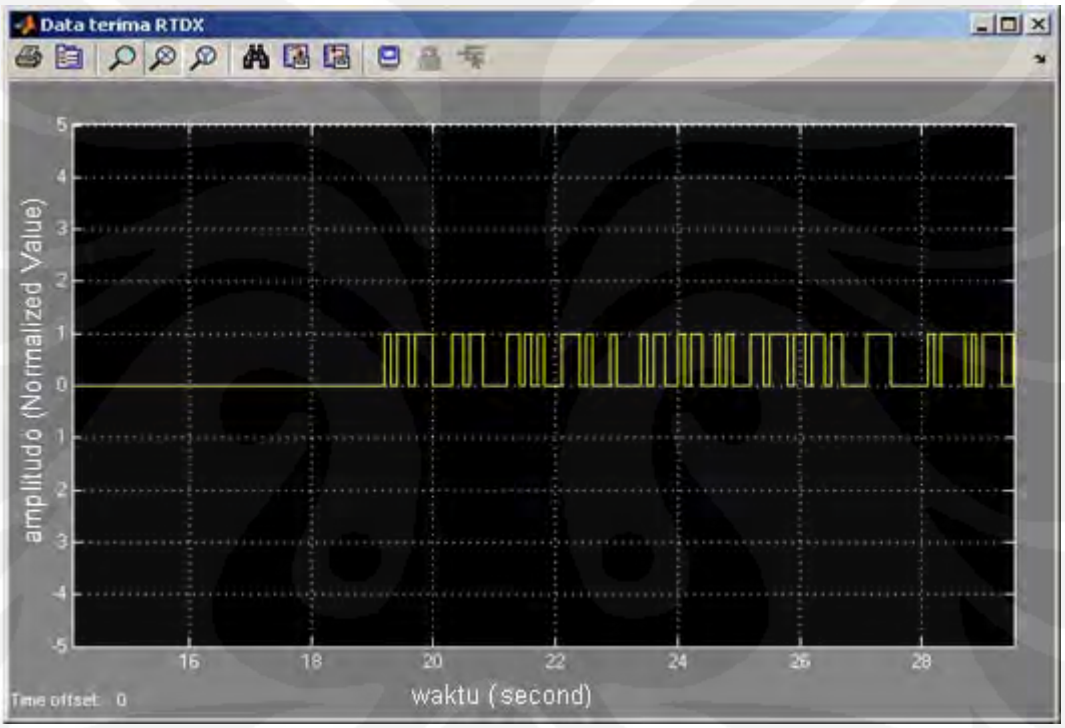
Pada ujicoba dengan menggunakan DSK TMS320C6713, pengujian dilakukan dengan menggunakan dua cara yaitu dengan menggunakan *Real Time Data Exchange* (RTDX) dan *storage osciloscope*.

4.2.1 *Real Time Data Exchange* (RTDX)

Ujicoba dilakukan dengan menggunakan RTDX karena *output* berupa bit-bit. Apabila *output* yang berupa bit-bit dilihat dengan menggunakan *storage osciloscope* maka *output* yang terlihat akan mengalami gangguan disebabkan oleh adanya proses sampling sinyal *digital* menjadi sinyal *analog* sedangkan jika

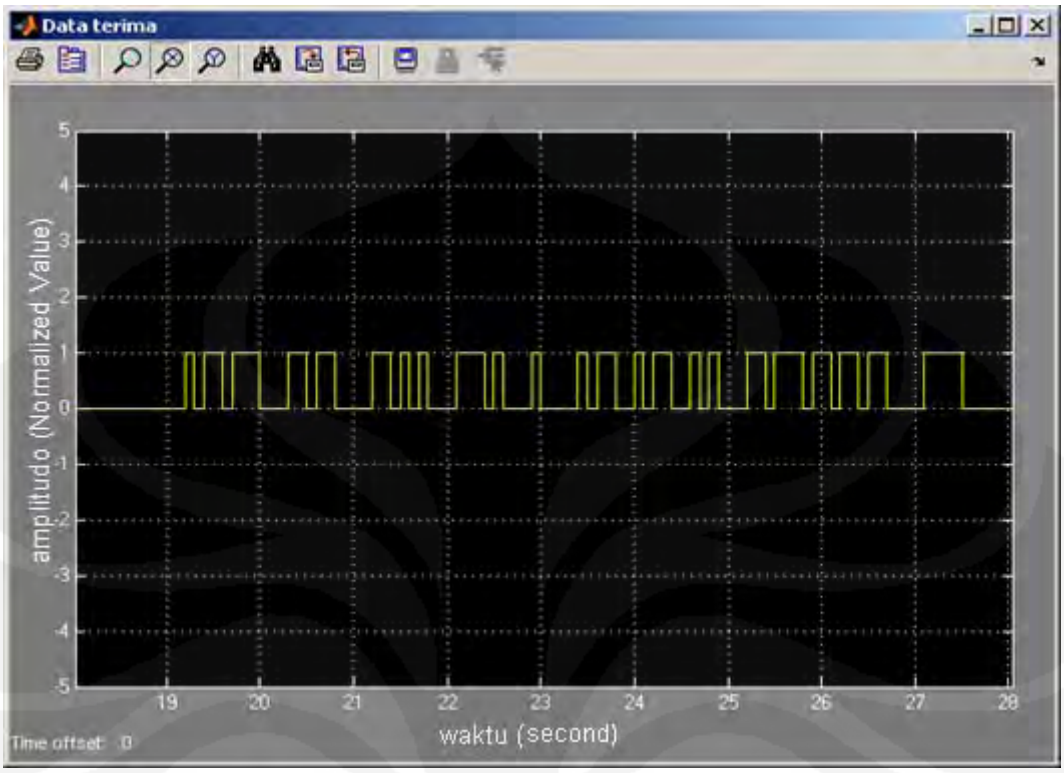
menggunakan RTDX maka *output* yang berupa bit-bit akan dikirim ke komputer dan data diolah secara *digital* sehingga tidak mengganggu atau merusak *output*.

Pengujian kebenaran data *output* dari *receiver* dilakukan dengan membandingkan data *output* RTDX dengan data *output* didapat dari simulink. *Input* pada pengujian ini sama dengan *input* pada pengujian simulink yakni *bernoulli binary generator* seperti dapat dilihat pada Gambar 4.1. *Output* dari rangkaian penerima OFDM dengan menggunakan RTDX didapatkan data seperti Gambar 4.7.



Gambar 4.7 *Output* rangkaian penerima OFDM dengan RTDX

Output dari rangkaian penerima OFDM yang didapat dari simulink dapat dilihat pada Gambar 4.8. Dari kedua gambar tersebut dapat diketahui bahwa data *output* yang didapat dengan menggunakan RTDX pada Gambar 4.7 sama dengan data *output* yang berasal dari simulink pada Gambar 4.8. Data *output* yang berasal dari simulink sama dengan data yang dikirim dari *transmitter*. Oleh karena itu, data *output* yang didapat dari RTDX sama dengan data yang dikirm dari *transmitter* pada Gambar 4.1. Hal ini menunjukkan bahwa rancang bangun model rangkaian penerima OFDM yang diuji sudah berfungsi dengan semestinya.

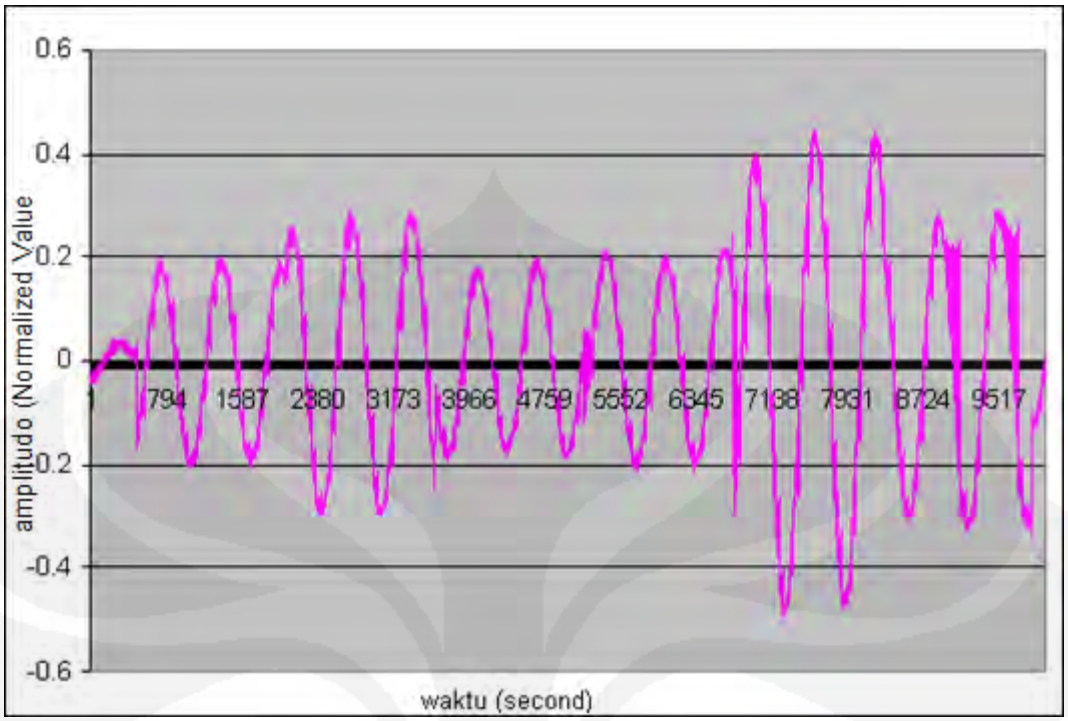


Gambar 4.8 *Output* rangkaian penerima OFDM

4.2.2 *Storage Osciloscope* Tektronix TDS 3052B

Ujicoba yang dilakukan pada bagian ini adalah ujicoba dengan DSK dan *output* diambil dengan menggunakan alat yang bernama *storage osciloscope* Tektronix TDS 3052B. Dengan menggunakan alat tersebut, data *output* bisa disimpan ke dalam disket dan hasilnya dapat dilihat dengan menggunakan komputer.

Pengambilan data dengan menggunakan *storage osciloscope* ini digunakan untuk mengambil data yang tidak berupa bit-bit tetapi berbentuk gelombang. Pada ujicoba ini data yang diambil adalah sinyal yang diterima *receiver* yang berasal dari *transmitter* OFDM. Bentuk gelombang OFDM dapat dilihat pada Gambar 4.9.



Gambar 4.9 Sinyal OFDM DSK

Pada Gambar 4.9 terlihat bahwa sinyal OFDM memiliki amplitudo dan sudut fasa yang berbeda-beda karena proses modulasi yang digunakan adalah modulasi QAM. Gambar 4.9 juga menunjukkan adanya *white noise* yang berpengaruh terhadap bentuk gelombang. Meskipun model rangkaian penerima OFDM tidak turut serta memperhitungkan *noise* yang terdapat pada kanal transmisi namun *noise* tetap ada dan berpengaruh.

Noise adalah sinyal yang tidak diinginkan dalam transmisi informasi. Sistem komunikasi yang baik memiliki nilai *noise* yang kecil. *White noise* adalah *noise* yang dihasilkan oleh pergerakan elektron secara *random* dalam media konduktif dan terdapat pada semua media transmisi dan peralatan komunikasi. Oleh sebab itu, *white noise* selalu menyertai sinyal informasi. *Noise* ini mempunyai distribusi energi yang seragam pada seluruh spektrum frekuensi. *White noise* berbanding lurus dengan *bandwidth* dan suhu. Persamaan matematis *white noise* dapat dilihat pada persamaan 4.1.

$$Pn\ (watt) = k\ T\ B \qquad (4.1)$$

Dari persamaan 4.1 dapat dinyatakan bahwa semakin besar *bandwidth* maka semakin besar pula *white noise*. *Noise* yang besar akan menjadikan nilai *signal to noise ratio* (SNR) menjadi kecil. Nilai SNR bisa didapatkan dengan menggunakan persamaan 4.2.

$$SNR\ (dB) = 10 \log (S/N) \qquad (4.2)$$

Dari persamaan 4.2 didapatkan bahwa hubungan antara SNR dengan *noise* adalah berbanding terbalik. Semakin kecil nilai *noise* maka semakin besar nilai SNR. Suatu sistem komunikasi dinyatakan baik apabila memiliki nilai SNR yang tinggi sehingga agar sistem komunikasi berjalan baik maka nilai *noise* harus dibuat sekecil mungkin.

BAB V
KESIMPULAN

Berdasarkan hasil analisis dapat disimpulkan :

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

DAFTAR ACUAN

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

[2] "*Orthogonal Frequency Division Multiplexing*", OFDM *tutorial*. Diakses 17 Maret 2008, dari complextoreal. http://www.complextoreal.com/chapters/ofdm2.pdf

[3] "Mengenal Teknologi *Frequency Division Multiplexing* (OFDM) pada Komunikasi *Wireless*", Elektro Indonesia, 1 Mei 2008. http://www.elektroindonesia.com/elektro/tel24.html

[4] Mahendra, A. "Petunjuk Penggunaan DSP TMS320C50", 12 September 2004.

[5] Frescura, F. "*Digital Signal Processors*", *Dipartimento di Ingegneria Elettronica e dell'Informazione Perugia*,1999.

[6] Poncius, Pontas., Herdian, Iwan."Modul DSP *Embedded*", RTMC-UI, Depok

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

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

[9] *Introduction to Simulink, Link for CCS & Real-Time Workshop*,1 Mei 2008. http://www.emba.uvm.edu/~mirchand/classes/EE275/2007/Real-Time/Lab6.pdf

DAFTAR PUSTAKA

Matlab *help, signal processing toolbox*, mathworks *inc.*

Matlab *help, communications blockset*, mathworks *inc.*

Mohan V, Arun., "*Orthogonal Frequency Division Multiplexing*", *seminar report, Department of Electronics and Communication Engineering*, 2000.

Bahai, Ahmad. R. S., Saltzberg, Burton. R, "*Theory and Applications of OFDM*", Kluwer *Academic Publisher*, New York: 1999.

Li, Ye., Stuber, Gordon, "*Orthogonal Frequency Division Multiplexing for Wireless Communications*", Springer, New York: 2006.

Schulze, Henrik., Luders, Christian, "*Theory and Applications of OFDM and CDMA*", John Wiley & Sons, Ltd, West Sussex: 2005.

Diponegoro, Arman Djohan., Suryanegara, Muhammad, "Rancang Bangun Rangkaian OFDM dengan Menggunakan DSK TMS C6713 Berbasis Simulink"

Ifeachor, E.C. dan Jervis, B.K., "*Digital Signal Processing*", Prentice Hall *Second Edition*, 2001

LAMPIRAN

Lampiran 1 Program RTDX

```
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=0.1;
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');
```