UNIVERSITAS INDONESIA

SIMULASI ERROR CORRECTION DENGAN PENGGABUNGAN TEKNIK REED-SOLOMON CODE (15,5) DAN BCH CODE (15,5) MENGGUNAKAN DSK TMS320C6713 BERBASIS SIMULINK

SKRIPSI

KHOTMAN HILMY FAJRIAN
0405030486

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2010

UNIVERSITAS INDONESIA

SIMULASI ERROR CORRECTION DENGAN PENGGABUNGAN TEKNIK REED-SOLOMON CODE (15,5) DAN BCH CODE (15,5) MENGGUNAKAN DSK TMS320C6713 BERBASIS SIMULINK

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana

KHOTMAN HILMY FAJRIAN
0405030486

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2010

HALAMAN PERNYATAAN ORISINALITAS

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

Nama	**:**	**KHOTMAN HILMY FAJRIAN**
NPM	**:**	**0405030338**
Tanda tangan	**:**	
Tanggal	**:**	**15 Juni 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Khotman Hilmy Fajrian
NPM	:	0405030486
Program Studi	:	Teknik Elektro
Judul Skripsi	:	Simulasi *Error Correction* Dengan Penggabungan Teknik Reed-Solomon Code (15,5) dan BCH Code (15,5) Menggunakan DSK TMS320C6713 Berbasis Simulink

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

DEWAN PENGUJI

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

Penguji : Dr. Ir. Arman D. Diponegoro ( )

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

Ditetapkan di : Depok
Tanggal : 7 Juni 2010

KATA PENGANTAR

Puji syukur saya panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Prof. Dr. Ir. Dadang Gunawan, M.Eng dan Dr. Ir. Arman D. Diponegoro, selaku dosen pembimbing 1 dan 2 yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan tugas akhir ini;
(2) Pihak laboratorium telekomunikasi Departemen Teknik Elektro yang telah banyak membantu dalam usaha memperoleh data dan peralatan yang saya perlukan;
(3) Orang tua saya, ibunda dan ayahanda yang terus tanpa pernah lelah memberikan yang segala terbaik untukku, tak henti memberikan dorongan menuju keberhasilan tanpa pernah menuntut. Semoga Allah Ta'ala senantiasa melimpahkan rahmatNya kepada ayah dan ibu. Kakak dan adik tercinta, yang selalu mendukung dan ingin melihat saudaranya menggapai keberhasilan;
(4) Teman setia penulis, Putri, yang tak pernah jemu mengingatkan penulis untuk tidak menyerah dalam menggapai cita-cita yang baik;
(5) Teman-teman elektro dan semua pihak yang telah membantu dalam pembuatan dan penyusunan tugas akhir ini.

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

Jakarta, 15 Juni 2010

Penulis

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

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

Nama : KHOTMAN HILMY FAJRIAN
NPM : 0405030486
Program Studi : Teknik Elektro
Departemen : Teknik Elektro
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Nonekslusif (*Non-exclusive Royalty-FreeRight*)** atas karya ilmiah saya yang berjudul:

SIMULASI ERROR CORRECTION DENGAN PENGGABUNGAN TEKNIK REED-SOLOMON CODE (15,5) DAN BCH CODE (15,5) MENGGUNAKAN DSK TMS320C6713 BERBASIS SIMULINK

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Nonekslusif ini, Universitas Indonesia berhak menyimpan, mengalihmedia/memformat-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Jakarta
Pada Tanggal : 15 Juni 2010
Yang menyatakan

(KHOTMAN HILMY FAJRIAN)

ABSTRAK

Nama : KHOTMAN HILMY FAJRIAN
Program Studi : Teknik Elektro
Judul : SIMULASI ERROR CORRECTION DENGAN PENGGABUNGAN TEKNIK REED-SOLOMON CODE (15,5) DAN BCH CODE (15,5) MENGGUNAKAN DSK TMS320C6713 BERBASIS SIMULINK

Skipsi ini membahas sebuah rancangan simulasi rangkaian pengoreksi kesalahan (*error correction*) dalam sistem komunikasi digital menggunakan program simulasi Simulink. Metode yang digunakan untuk mendeteksi dan mengoreksi kesalahan adalah metode penyandian siklis dengan teknik penyandian BCH (Bose Chaudhury Hocquenhem) C_{BCH}(n,k) dan RS (Reed-Solomon) C_{RS}(n,k). Kedua teknik tersebut digabungkan secara serial (*concatetaned*), dimana panjang kata sandi yang dipergunakan untuk penyandian RS adalah 15 simbol (digit) dan untuk BCH 15 bit. Sedangkan panjang informasinya, 5 simbol untuk penyandian RS dan kelipatan bulat k bit untuk BCH. Teknik RS (15,5) dapat memperbaiki kesalahan sebanyak 5 simbol secara acak, sedangkan teknik BCH (15,5) mampu memperbaiki 3 bit kesalahan secara acak. Dengan penggabungan kedua teknik tersebut dihasilkan rangkaian *error correction* yang mampu mengoreksi kesalahan acak sekitar 36 *bit* dan kesalahan mengelompok hingga 116 *bit* di belakang dan 43 *bit* di mana saja.

Rancangan simulasi penyandian ini kemudian diaplikasikan pada perangkat *signal processing* DSK6713 dengan cara memasukkan rancangan program simulink ke perangkat DSK tersebut melalui program *C6000 Code Composer Studio* versi 3.1. Dari hasil pengujian dengan menggunakan fasilitas RTDX didapat bahwa program yang dibuat dengan model simulink dapat berjalan dengan baik pada DSK6713 dan sesuai dengan simulasi pada program simulink.

Kata Kunci:
RS, BCH, Simulink, DSK6713, RTDX

ABSTRACT

Name : KHOTMAN HILMY FAJRIAN
Study Program : Electrical Engineering
Title : SIMULATION OF ERROR CORRECTION BY COMBINING REED-SOLOMON CODE (15,5) AND BCH CODE (15,5) USING DSK TMS320C6713 BASED ON SIMULINK MODEL

This thesis make a simulation design of an equipment for error correction in digital communication system using simulation program Simulink. The methode used to detect and correct the error is cyclic coding methode with BCH (Bose Chaudhury Hocquenhem) Code $C_{BCH}(n,k)$ and RS (Reed-Solomon) Code $C_{RS}(n,k)$ technique. Both the technique are combined in serial configuration (concatenated). For RS code we use 15 symbols (digits) codeword and 5 symbols information length. Whereas for BCH code we use 15 bits codeword and integer multiple of k bits information length. RS code (15,5) can correct 5 symbols random error and BCH code (15,5) can correct 3 bits random error. By combining the techniques obtained an error correction circuit that can correct up to about 36 bits of random error and up to 116 bits in rear and 43 bits anywhere of burst error.

Simulation design of this coding then will be aplicated on signal processing equipment of DSK6713 by injecting the simulation program designed by simulink to the DSK using program C6000 Code Composer Studio version 3.1. From the result of the simulation testing using RTDX obtained that the program designed by simulink can be executed properly on DSK6713 and is suitable with the result of simulation on simulink.

Keyword:
RS, BCH, Simulink , DSK6713, RTDX

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB 1
PENDAHULUAN

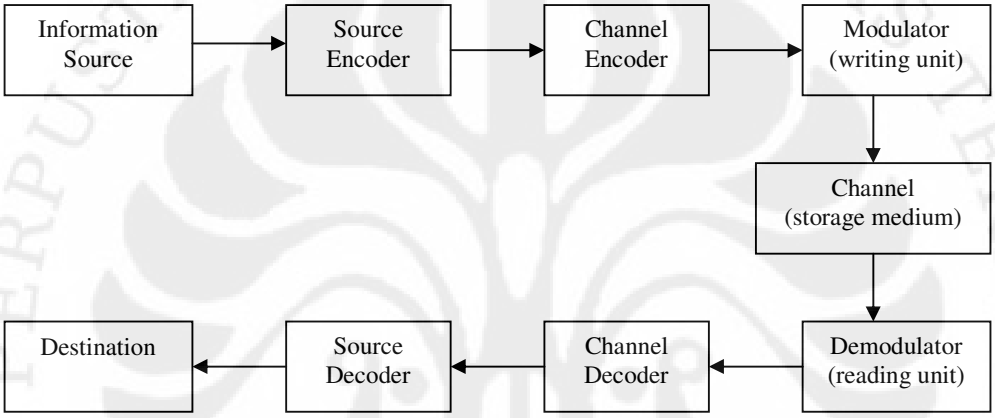
1.1 LATAR BELAKANG MASALAH

Di tengah perkembangan sistem komunikasi digital yang begitu pesat, kemajuan di bidang teknologi memegang peranan penting. Di antara yang mendorong begitu cepatnya perkembangan teknologi komunikasi digital adalah adanya tuntutan kebutuhan akan suatu sistem perngiriman data digital dan sistem penyimpanan data yang efisien dan handal. Selain itu juga dipercepat oleh mendesaknya kebutuhan akan sistem komunikasi data dengan skala besar dan kecepatan tinggi di berbagai lingkup kehidupan. Namun demikian, semaju apapun teknologi sistem komunikasi digital, tetap ada faktor yang terlibat dalam sistem tersebut yang menyebabkan tidak dapat dicapainya suatu kondisi yang ideal. Faktor tersebut berupa gangguan dalam saluran transmisi atau media penyimpanan data yang menyebabkan informasi yang diterima atau dibaca dari media penyimpanan mengalami perubahan atau kerusakan. Gangguan tersebut dapat berupa derau atau interferensi pada saluran transmisi, dan tidak sempurna atau cacatnya media penyimpanan.

Untuk mempertahankan informasi yang dikirim hingga sampai ke pengguna diperlukan suatu mekanisme yang dapat melindungi informasi tersebut, yang salah satunya adalah dengan menggunakan kata sandi atau penyandian terhadap informasi yang hendak dikirim (*error control codes*). Dengan metode penyandian ini data yang diterima dapat diterjemahkan kembali ke bentuk informasi aslinya meskipun telah mengalami perubahan atau kerusakan karena penyandian ini berfungsi untuk memperbaiki kesalahan yang terjadi pada informasi digital. Gambar 1.1 menujukkan sebuah diagram blok sistem transmisi atau penyimpanan data yang menerapkan teknik penyandian untuk meminimalkan dampak gangguan dalam saluran atau media penyimpanan.

Untuk menghasilkan sebuah *error control codes*, ada beberapa macam teknik penyandian salah satunya yaitu penyandian jenis siklis (*cyclic code*). Penyandian siklis memiliki kelebihan dibandingkan dengan jenis penyandian lainnya di antaranya adalah dapat mendeteksi dan mengoreksi adanya kesalahan

data yang diterima sekaligus. Teknik penyandian yang termasuk dalam jenis siklis dan banyak dikembangkan untuk memperbaiki kemampuannya di dalam mengoreksi kesalahan, di antaranya RS (*Reed-Solomon*), BCH (*Bose Chaudhuri Hocquenhem*), dan Meggit. Penggabungan teknik yang ada memungkinkan untuk melengkapi karakter dan kemampuan masing-masing teknik dalam memperbaiki kesalahan yang lebih bervariasi. Penggunaan program simulink memungkinkan penggabungan kedua teknik yaitu RS dan BCH untuk dirancang dan disimulasikan untuk selanjutnya rancangan tersebut diterapkan pada perangkat DSK TMS320C6713.



Gambar 1.1 Diagram blok sistem transmisi atau penyimpanan data [1]

1.2 PERUMUSAN MASALAH

Berdasarkan latar belakang tersebut, dapat dirumuskan masalah yaitu bagaimana merancang rangkaian sistem *error correction* untuk komunikasi digital dengan penggabungan dua penyadian yang diterapkan pada perangkat DSP. Rumusan masalah dapat diperinci menjadi tiga pertanyaan sebagai berikut:

1. Bagaimana penggabungan teknik penyandian BCH dan RS dapat meningkatkan kemampuan memperbaiki kesalahan informasi?
2. Bagaimana merancang gebungan teknik penyandian BCH dan RS menggunakan program simulink?
3. Bagaimana menerapkan rancangan dari program simulink ke dalam perangkat DSK?

1.3 TUJUAN

Skripsi ini bertujuan untuk merancang bangun rangkaian simulasi yang berfungsi memperbaiki kesalahan dalam komunikasi digital dengan menggabungkan dua teknik penyandian yaitu penyandian RS (15,5) dan penyandian BCH (15,5) melalui program simulink yang diaplikasikan pada perangkat DSK TMS320C6713.

1.4 BATASAN MASALAH

Perancangan perangkat *error correcting* ini menggunakan dua jenis penyandian siklis yaitu penyandian RS (15,5) dan BCH (15,5) secara berurutan dengan pesan awal sepanjang 5 simbol *integer*. Panggabungan kedua teknik penyandian tersebut dilakukan dengan konfigurasi sesial menggunakan program simulink. Hasil perancangan menggunakan program simulink tersebut kemudian diterapkan pada perangkat DSK TMS320C6713, tanpa membahas tentang teknik modulasi yang digunakan dalam simulasi dan pengujian DSK hanya menggunakan fasilitas RTDX untuk memperoleh data dari DSK.

1.5 SISTEMATIKA PENULISAN

Penulisan tugas akhir ini disusun berdasarkan sistematika sebagai berikut.

Bab 1 Pendahuluan

Berisi latar belakang masalah, perumusan masalah, tujuan penulisan, batasan masalah, dan sistematika penulisan.

Bab 2 Penyandian siklis dan DSK TMS320C6713

Berisi tentang prinsip dasar dari penyandian blok siklis secara umum, penyandian BCH dan Reed-Solomon serta DSK TMS320C6713.

Bab 3 Perancangan simulasi

Berisi mekanisme perancangan simulasi baik menggunakan simulink maupun perangkat DSK TMS320C6713.

Bab 4 Pengujian dan analisis sistem

Berisi pengujian rancangan simulasi sistem *error correction* dengan simulink dan DSK TMS320C6713 serta analisis terhadap sistem.

Bab 5 Kesimpulan

Berisikan kesimpulan yang didapat dari tugas akhir ini.

BAB 2
PENYANDIAN SIKLIS DAN DSK TMS320C6713

2.1 KONSEP PENYANDIAN SIKLIS

Penyandian siklis merupakan salah satu jenis penyandian blok linier. Penyandian ini menggunakan struktur matematika aljabar linier dengan bentuk polinomial pada pembentukan kata sandi dan penguraian kata sandi, sedangkan penyandian lainnya menggunakan matriks dalam pembentukan kata sandi dan penguraiannya sehingga penyandian model siklis lebih mudah dalam melakukan penyandian dan penguraian dibandingkan jenis penyandian blok lainnya. Namun demikian, pembentukan kata sandi pada penyandian siklis juga dapat dilakukan dengan operasi matriks yang merupakan operasi dasar dari pembentukan kata sandi pada penyandian blok.

Suatu penyandian biner dikatakan sebagai penyandian siklis jika memiliki 2 sifat dasar, yaitu [1]:

1. Sifat linier, artinya yaitu bila 2 buah kata sandi dijumlahkan maka hasilnya juga merupakan sebuah kata sandi.
2. Sifat siklis, artinya pergeseran siklis dari suatu kata sandi adalah kata sandi.

Sifat linier pada penyandian siklis menegaskan bahwa penyandian siklis adalah pernyandian blok linier. Misalkan c = $(c_0, c_1, c_2, ..., c_{n-1})$ adalah suatu kata sandi dari penyandian blok linier (n,k), jika dilakukan pergeseran siklis maka kata sandi dengan n digit berbentuk $(c_{n-1}, c_0, c_0, ..., c_{n-2})$, $(c_{n-2}, c_0, c_0, ..., c_{n-1})$, $(c_1, c_2, c_3, ..., c_0)$. Contoh tersebut menunjukkan sifat siklis yang dimiliki penyandian siklis. Sifat siklis ini juga menandakan bahwa unsur pembentuk kata sandi dengan panjang n dapat dianggap sebagai koefisien dari suatu polinomial dengan pangkat (n-1), sehingga kata sandi dengan unsur pembentuk $(c_0, c_1, c_2, ..., c_{n-1})$ dapat dinyatakan dalam bentuk polinomial kata sandi berikut [3]:

$$c(x) = (c_0 + c_1 x + ... + c_{n-1} x^{n-1}) \quad (2.1)$$

dan bila digeser siklis dapat ditulis:

$$c^{(i)}(x) = c_{n-1} + c_{n+1-i} x + ... + c_{n-1} x^{i-1} + c_0 x^i + c_1 x^{i+1} + ... + c_{n-i-1} x^{n-1} \quad (2.2)$$

dimana:

i = jumlah pergeseran

n = jumlah bit dalam satu blok

c = dapat bernilai 0 atau 1

Jika $x^{n-1} = 1$ maka tiap pangkat x dari polinomial c(x) mewakili satu pergeseran siklis, sehingga perkalian polinomial c(x) dengan x dipandang sebagai pergeseran siklis atau satu putaran ke kanan. Bentuk perkalian polinomial ini merupakan modulo $(x^n - 1)$, dimana untuk satu pergeseran siklis [2]:

$$xc(x) \bmod (x^n - 1) = c_{n-1} + c_0 x + \ldots + c_{n-2} x^{n-1} \quad (2.3)$$

2.2 POLINOMIAL GENERATOR DAN *PARITY CHECK* PADA PENYANDIAN SIKLIS

Polinomial generator siklis adalah suatu polinomial yang digunakan dalam perhitungan pembentukan kata sandi dan penguraiannya dengan berdasarkan pada faktor penyandian siklis (n,k) yang dispesifikasikan oleh polinomial kata sandi berderajat (n-1) atau kurang, dan memiliki polinomial berderajat (n-k) sebagai faktornya. Polonomial ini dinyatakan dengan g(x). Derajat g(x) menunjukkan banyaknya bit pariti polonomial generator g(x) ekivalen dengan matriks generator G pada penyandian blok linier.

Polinomial pariti pemeriksa (*parity check*) dibentuk oleh polinomial yang berderajat k dan disebut h(x). Polinomial *parity check* merupakan bentuk ekivalen dari matriks penyandian blok linier sesuai dengan hubungan [2]:

$$h(x) g(x) \bmod (x^n - 1) = 0 \quad (2.4)$$

Sesuai dengan sifat dasar siklis bahwa setiap kelipatan polinomial generator g(x) dari pengandian siklis (n,k) adalah polinomial kata sandi, maka dari persamaan (2.4) diperoleh bahwa setiap polinomial kata sandi c(x) memenuhi persamaan[2]:

$$h(x) c(x) \bmod (x^n - 1) = 0 \quad (2.5)$$

Polinomial generator dan polinomial *parity check* memiliki sifa-sifat dasar yaitu:

1. Polinomial generator penyandian siklis (n,k) adalah satu-satunya polinomial kata sandi yang berderajat minimum (n-k) dan dapat ditulis:

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k-1} x^{n-k-1} + x^{n-k} \quad (2.6)$$

Dimana g_0 = 1, agar syarat derajat minimum tercapai.

2. Perkalian polinomial g(x) dengan satu polinomial p(x) adalah polinomial kata sandi juga bila p(x) adalah polinomial dalam x. Sehingga dapat ditulis:

$$c(x) = p(x)g(x) \bmod (x^n - 1) \tag{2.7}$$

3. Polinomial generator g(x) dan polinomial *parity check* h(x) adalah faktor dari polinomial ($x^n + 1$) sesuai dengan persamaan:

$$h(x)g(x) = x^n + 1 \tag{2.8}$$

Dimana perhitungan aritmatika modulo-2, nilai ($x^n + 1$) sama dengan ($x^n - 1$) dan perkalian g(x) dengan p(x) dalam bentuk yang diperluas:

$$p(x)g(x) = p_0 g(x) + p_1 g(x) + ... + p_{k-1} x^{k-1} g(x) \tag{2.9}$$

Dari sifat dasar siklis pertama diketahui $x^n g(x)$ adalah polinomial kata sandi, oleh sebab itu dapat disimpulkan bahwa setiap perkalian dari g(x) modulo ($x^n - 1$) adalah kata sandi juga seperti pada persamaan (2.9). Kesimpulan ini dapat dinyatakan bahwa satu polinomial biner berderajat n-1 atau kurang adalah polinomial kata sandi, jika dan hanya jika polinomial tersebut adalah kelipatan g(x).

Berdasarkan sifat dasar siklis, khususnya sifat ketiga, maka dapat dikatakan bahwa bila g(x) adalah polinomial berderajat (n-k) dan merupakan faktor dari $x^n + 1$, maka g(x) adalah polinomial generator dari penyandian siklis (n,k). Seperti pada g(x), maka h(x) adalah polinomial *parity check* penyandian siklis (n,k) bila h(x) adalah polinomial berderajat k dan merupakan faktor dari $x^n + 1$. Hal inilah yang menjadi dasar pemilihan polinomial generator dan polinomial *parity check*.

2.3 PROSES PENYANDIAN (*ENCODING*)

Proses penyandian pada penyandian siklis bertujuan untuk membentuk kata sandi pada sisi pengirim untuk dikirimkan melalui sebuah kanal. Pembentukan kata sandi berdasarkan pada polinomial generator g(x) yang dipilih. Agar didapat penyandian siklis yang sistematis, blok informasi terpisah dari blok pariti dan harus memenuhi struktur berikut[3]:

$$(h_0, h_1, h_2, ..., h_{n-k-1}, m_0, m_1, m_2, ..., m_{k-1})$$

dimana h_0 hingga h_{n-k-1} adalah *bit-bit* pariti dan m_0 hingga m_{k-1} adalah *bit-bit* informasi.

Jika $m(x)=m_0+m_1x+...+m_{k-1}x^{k-1}$ adalah polinomial informasi, maka untuk memenuhi struktur sistematis polinomial m(x) dikalikan dengan x^{n-k}, maka diperoleh:

$$x^{n-k}m(x)=m_0x^{n-k}+m_1x^{n-k+1}+...+m_{k-1}x^{n-1} \quad (2.10)$$

Polinomial pariti h(x) adalah sisa pembagian terhadap g(x). Maka persamaan pembentukan kata sandi sistematis adalah:

$$x^{n-k}m(x)=v(x)g(x)+h(x) \quad (2.11)$$

dimana v(x) adalah hasil pembagian.

Persamaan (2.11) dapat dituliskan kembali menjadi:

$$h(x)+x^{n-k}m(x)=v(x)g(x) \quad (2.12)$$

Pada persamaan (2.12) polinomialnya merupakan kelipatan dari polinomial generator g(x), maka polinomial itu adalah polinomial kata sandi dari penyandian siklis (n,k) yang dihasilkan oleh g(x). Hal ini dapat ditunjukkan pada persamaan:

$$c(x)=h(x)+x^{n-k}m(x) \quad (2.13)$$

Maka untuk membentuk kata sandi sistematis, penyandian siklis (n,k) dilakukan tiga tahap, yaitu [1]:

1. Polinomial informasi m(x) dikalikan dengan x^{n-k}.
2. Bagi $x^{n-k}m(x)$ dengan polinomial generator g(x) untuk menghasilkan sisa h(x).
3. Polinomial kata sandi c(x) adalah penambahan h(x) terhadap $x^{n-k}m(x)$.

2.4 PROSES PENGURAIAN (*DECODING*)

Proses penguraian kata sandi pada dasarnya melakukan perhitungan sindrom (*syndrome*). Perhitungan sindrom diperlukan untuk memeriksa apakah kata sandi yang diterima mengalami kerusakan atau perubahan ataukah tidak. Misalnya kata sandi $c=(c_0,c_1,c_2,....,c_{n-1})$ ditransmisikan dan mengalami gangguan sehingga menghasilkan kata sandi yang diterima $r=(r_0,r_1,r_2,....,r_{n-1})$ dengan polinomial yang diterima r(x). Maka terdapat unsur kesalahan e atau polinomial e(x) pada kata sandi yang diterima r(x), sehingga persamaan polinomial dari kondisi ini [1]:

$$r(x) = c(x) + e(x) \tag{2.14}$$

Polinomial sindrom s(x) merupakan sisa pembagian r(x) terhadap g(x) dan dapat dituliskan sebagai persamaan:

$$s(x) = r(x) \bmod g(x) = s_0 + s_1 x + \ldots + s_{n-k-1} x^{n-k-1} \tag{2.15}$$

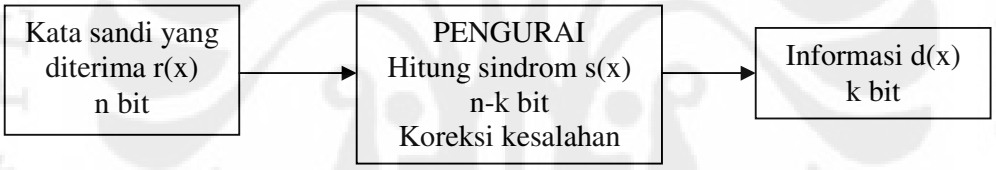
Jika terjadi kesalahan atau timbulnya polinomial e(x) maka $s(x) \neq 0$, tetapi jika tidak terjadi kesalahan mana s(x) = 0 dan e(x) = 0. Jika persamaan (2.14) dihubungkan dengan persamaan (2.15), maka e(x) menjadi:

$$e(x) = c(x) + a(x)g(x) + s(x) \tag{2.16}$$

dimana $a(x)$ adalah hasil bagi r(x) terhadap g(x).

Dari persamaan (2.16) dapat dilihat tujuan penguraian adalah untuk memperkirakan polinomial kesalahan e(x) dari polinomial sindrom.

Alur penguraian kata sandi yang diterima pada penyandian siklis dapat dilihat pada Gambar 2.1 berikut:



Gambar 2.1 Alur Penguraian Kata Sandi Dengan Penyandian Siklis di Sisi Penerima

2.5 PENYANDIAN DAN PENGURAIAN PADA TEKNIK PENYANDIAN SIKLIS BCH DAN RS

Penyandi (*encoder*) dan pengurai (*decoder*) pada penyandian siklis banyak dikembangkan dengan teknik-teknik khusus yang bertujuan meningkatkan kemampuan memperbaiki kesalahan kata sandi. Panjang informasi yang akan disandikan sangat menentukan tingkat perbaikan kesalahan yang dapat dilakukan oleh teknik penyandian siklis. Dari beberapa teknik penyandian yang ada, terdapat dua teknik yang memiliki keunggulan dalam tingkat perbaikan kesalahannya. Kedua teknik itu adalah Bose Chaudhury Hocquenghem (BCH) dan Reed-Solomon (RS).

2.5.1 Bose Chaudhury Hocquenghem (BCH)

Pada tahun 1959 Hocquenghem merepresentasikan suatu metode penyandian siklis yang memiliki tingkat perbaikan kesalahan yang tinggi dengan pembentukan kata sandi yang lebih spesifik. Kemudian pada tahun 1960 metode tersebut disempurnakan oleh Bose dan Chaudhury, sehingga teknik penyandian ini disebut Bose Chaudhury Hocquenghem atau BCH.

2.5.1.1 Polinomial Generator

Perhitungan pada penyandian BCH melibatkan perhitungan aritmatika *Galois Field* (GF), yaitu suatu area dari dua elemen biner yang operasi penjumlahan dan perkaliannya didefinisikan pada modulo-2 dan dinyatakan sebagai GF (2^m). *Galois Field* sangat berhubungan dengan panjang kata sandi (n), dan dapat dituliskan dengan persamaan:

$$n = 2^m - 1 \tag{2.17}$$

Kemampuan memperbaiki kesalahan pada teknik BCH dinyatakan sebagai t dengan persamaan[1]:

$$t \geq \frac{n-k}{m} \tag{2.18}$$

dimana t menunjukkan jumlah bit yang mampu diperbaiki dalam satu blok.

Jika n = 15 pada BCH (15,5) disubstitusikan ke dalam persamaan (2.17) akan diperoleh nilai m = 4 atau GF (2^4). Kemudian nilai-nilai yang ada disubstitusikan ke dalam persamaan (2.18), sehingga dipeoleh $t \geq \frac{15-5}{4} \approx 3$, yang berarti kemampuan perbaikan dari BCH (15,5) adalah 3 digit. Untuk penyandian dengan GF (2^4) dibangkitkan polinomial primitif $p(x) = 1 + x + x^4$, dimana p(x) adalah polinomial primitif untuk GF(2^4) yang tidak dapat difaktorkan lagi. Tabel 2.1 menunjukkan polinomial primitif untuk GF(2^m) dari nilai m = 2 sampai dengan m = 27.

Tabel 2.1 Polinomial Primitif Pada GF(2)[1]

m	p(x)	m	p(x)
2	$x^2 + x + 1$	**15**	$x^{15} + x + 1$
3	$x^3 + x + 1$	**16**	$x^{16} + x^{12} + x^3 + x + 1$
4	$x^4 + x + 1$	**17**	$x^{17} + x^3 + 1$

5	$x^5 + x^2 + 1$	**18**	$x^{18} + x^7 + 1$
6	$x^6 + x + 1$	**19**	$x^{19} + x^5 + x^2 + x + 1$
7	$x^7 + x^2 + 1$	**20**	$x^{20} + x^3 + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$	**21**	$x^{21} + x^2 + 1$
9	$x^9 + x^4 + 1$	**22**	$x^{22} + x + 1$
10	$x^{10} + x^3 + 1$	**23**	$x^{23} + x^5 + 1$
11	$x^{11} + x^2 + 1$	**24**	$x^{24} + x^7 + x^2 + x + 1$
12	$x^{12} + x^6 + x^4 + x + 1$	**25**	$x^{25} + x^3 + 1$
13	$x^{13} + x^4 + x^3 + x + 1$	**26**	$x^{26} + x^6 + x^2 + x + 1$
14	$x^{14} + x^{10} + x^6 + x + 1$	**27**	$x^{27} + x^5 + x^2 + x + 1$

Polinomial generator untuk tingkat kemampuan perbaikan t pada BCH adalah polinomial berderajat rendah dari GF(2) yang memiliki $(\alpha, \alpha^2, \alpha^3, ..., \alpha^{2t})$ untuk $g(\alpha^i) = 0$ setiap akar-akarnya, dimana nilai untuk $1 \le i \le 2t$. Polinomial g(x) dibentuk berdasarkan polinomial minimal m(x). Polinomial minimal m(x) ditentukan untuk setiap g(x) yang memiliki koefisien dan akar-akarnya saling berkonjugasi. Maka polinomial generator dengan tingkat kemampuan perbaikan t dibentuk dari gabungan perkalian terkecil LCM (*Least Common Multiple*) dari $\phi_1(x), \phi_2(x), ..., \phi_{2t}(x)$, dan dinyatakan dengan [1]:

$$g(x) = LCM\{\phi_1(x), \phi_2(x), ..., \phi_{2t}(x)\} \quad (2.19)$$

Tabel 2.2 Polinomial Minimal dari Elemen-elemen GF(2^4) yang dibangkitkan oleh $p(x) = 1 + x + x^4$ [3]

Konjugat	**Polinomial minimal**
0	x
1	$1 + x$
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$1 + x + x^4$
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$1 + x + x^2 + x^3 + x^4$
α^5, α^{10}	$1 + x + x^2$
$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$1 + x^3 + x^4$

Dengan menggunakan tabel 2.2 di atas maka persamaan (2.19) dapat disederhanakan menjadi persamaan [1]:

$$g(x) = LCM\{\phi_1(x), \phi_3(x), ..., \phi_{2t-1}(x)\} \quad (2.20)$$

Dari persamaan (2.18) dan persamaan (2.20) dapat dibentuk polinomial generator untuk BCH (15,5) dengan t = 3, yaitu:

$$g(x) = LCM\{\phi_1(x), \phi_3(x), \phi_5(x)\}$$

$$g(x) = (1 + x + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2)$$

$$g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$$

Derajat polinomial tertinggi dari generator yang dihasilkan dari persamaan (2.20) adalah 10. Hasil ini sesuai dengan teori untuk membentuk kata sandi dengan n = 15 dan panjang informasi k = 5 diperlukan derajat polinomial tertinggi dari generator tersebut adalah :

$$|\deg(g(x))| = 15 - 5 = 10$$

Untuk melakukan penyandian dibentuk suatu konstruksi dari perhitungan aritmatika dengan suatu nilai α yang mewakili posisi biner dari kombilasi biner 2^3 seperti diberikan pada Tabel 2.3.

Tabel 2.3 Representasi α dari elemen GF(2^4) dibangkitkan oleh $p(x) = 1 + x + x^4$ [3]

Representasi Eksponen	**Representasi Polinomial**	**Representasi vektor**			
0	**0**	**0**	**0**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**
α	α	**0**	**1**	**0**	**0**
α^2	α^2	**0**	**0**	**1**	**0**
α^3	α^3	**0**	**0**	**0**	**1**
α^4	$1+\alpha$	**1**	**1**	**0**	**0**
α^5	$\alpha+\alpha^2$	**0**	**1**	**1**	**0**
α^6	$\alpha^2+\alpha^3$	**0**	**0**	**1**	**1**
α^7	$1+\alpha+\alpha^3$	**1**	**1**	**0**	**1**
α^8	$1+\alpha^2$	**1**	**0**	**1**	**0**
α^9	$\alpha+\alpha^3$	**0**	**1**	**0**	**1**
α^{10}	$1+\alpha+\alpha^2$	**1**	**1**	**1**	**0**
α^{11}	$\alpha+\alpha^2+\alpha^3$	**0**	**1**	**1**	**1**
α^{12}	$1+\alpha+\alpha^2+\alpha^3$	**1**	**1**	**1**	**1**
α^{13}	$1+\alpha^2+\alpha^3$	**1**	**0**	**1**	**1**
α^{14}	$1+\alpha^3$	**1**	**0**	**0**	**1**

2.5.1.2 Penyandian

Proses pembentukan kata sandi pada teknik yang digunakan dalam penyandian siklis bergantung dari aturan-aturan yang digunakan dalam masing-masing teknik. Pada BCH, proses pembentukan kata sandi mengikuti prinsip dasar proses pembentukan kata sandi pada penyandian siklis.

Jika pada BCH (15,5) panjang kata sandi n = 15, panjang informasi k = 5, dan $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$, maka dapat dibentuk kata sandi dengan menggunakan persamaan (2.11) dan (2.13):

$$h(x) + x^{n-k} m(x) = v(x) g(x)$$

$$c(x) = h(x) + x^{n-k} m(x)$$

dimana :

m(x) adalah polinomial informasi,

g(x) adalah polinomial generator,

v(x) adalah polinomial hasil pembagian $x^{n-k}d(x)$ terhadap g(x),

h(x) adalah polinomial sisa pembagian, dan

c(x) adalah polinomial kata sandi.

2.5.1.3 Penguraian

Perhitungan Sindrom

Jika kata sandi yang diterima merupakan kata sandi yang ditambah dengan polinomial kesalahan e(x), maka sindrom akan dihitung dari polinomial kata sandi yang diterima r(x). Pada BCH dengan tingkat kemampuan perbaikan t, banyaknya sindrom q ditentukan melalui persamaan sebagai berikut[1]:

$$q = 2t \tag{2.21}$$

Masing-masing terdiri dari 2 bit. Untuk BCH (15,5) dengan t = 3, jumlah sindrom q = 6 atau $(S_1, S_2, S_3, S_4, S_5, S_6)$. Sehingga perhitungan sindrom dapat dilakukan sebagai berikut [2]:

$$Si = r(\alpha^i)$$

$$= r_0 + r_1\alpha^i + r_2\alpha^{2i} + ... + r_{14}\alpha^{14i} \tag{2.22}$$

karena polinomial r(x) dibentuk oleh polinomial minimal $\phi_i(x)$ dari α^i, maka:

$$r(x) = \alpha_i(x)\phi_i(x) + b_i(x) \tag{2.23}$$

koefisien $b_i(x)$ menyatakan sisa dengan derajat yang tidak kurang dari $\phi_i(x)$ dan karena $\phi_i(x) = 0$, maka diperoleh[1]:

$$Si = r(\alpha^i) = b_i(\alpha^i) \tag{2.24}$$

Dari sindrom yang diperoleh, pengurai akan mencari polinomial pola kesalahan e(x) untuk menentukan letak kesalahan yang terdapat pada suatu kata sandi r(x) dengan jangkauan dari x^0 sampai x^{n-1}. Karena $\alpha, \alpha^2, \alpha^3, ..., \alpha^{2t}$ adalah akar-akar dari polinomial kata sandi, dimana $c_i(x) = 0$ untuk $1 \le i \le 2t$, sehingga dapat dinyatakan hubungan sindrom dan pola kesalahan:

$$Si = e(\alpha^i) \tag{2.25}$$

Dimana pola kesalahan e(x) memiliki w kesalahan pada lokasi $x^{j1}, x^{j2}, ..., x^{jw}$ membentuk $e(x) = x^{j1}, x^{j2}, ..., x^{jw}$ dan $S_{2t} = (\alpha^{j1})^{2t}, (\alpha^{j2})^{2t}, ..., (\alpha^{jw})^{2t}$ dengan nilai-nilai $\alpha^{j1}, \alpha^{j2}, ..., \alpha^{jw}$ sebagai koefisien-koefisien α yang menyatakan posisi biner yang tidak diketahui.

Pada BCH (15,7) dengan kemampuan perbaikan t = 3 atau kurang, sindrom yang akan dihitung adalah S_1 sampai dengan S_6 dan dapat dituliskan[1]:

$$\begin{aligned} S_1 &= \mathbf{r}(\alpha) = e_{j_1}\beta_1 + e_{j_2}\beta_2 + \cdots + e_{j_v}\beta_v \\ S_2 &= \mathbf{r}(\alpha^2) = e_{j_1}\beta_1^2 + e_{j_2}\beta_2^2 + \cdots + e_{j_v}\beta_v^2 \\ &\vdots \\ S_{2t} &= \mathbf{r}(\alpha^{2t}) = e_{j_1}\beta_1^{2t} + e_{j_2}\beta_2^{2t} + \cdots + e_{j_v}\beta_v^{2t} \end{aligned} \tag{2.26}$$

dimana v adalah jumlah lokasi kesalahan.

Penentuan Polinomial Pola Kesalahan (e(x))

Teknik penentuan lokasi kesalahan pada BCH dengan kondisi [4]:

1. untuk 1 kesalahan

   $\tau = S_1$

2. untuk 2 kesalahan

   $\tau_1 = S_1$

   $\tau_2 = \dfrac{S_3 + S_1^3}{S_1}$

3. untuk 3 kesalahan

$$\tau_1 = S_1$$

$$\tau_2 = \frac{S_1^{\,2} S_3 + S_5}{S_3 + S_1^{\,3}}$$

$$\tau_3 = S_3 + S_1^{\,3} + S_1 \tau_2$$

2.5.2 Reed-Solomon (RS)

Pada Desember 1958 Irving S. Reed dan Gustavo Solomon menyelesaikan laporan berjudul "*polunomial codes over certain finite fields"* pada M.I.T. Lincoln Laboratory. Kemudian tahun 1960 dilakukan sedikit modifikasi pada laporan tersebut dan diumumkan pada sebuah jurnal Society for Industrial and Applied Mathematics (SIAM) dan dikenal sebagai Reed-Solomon Codes yang merupakan sandi terbaru dalam perbaikan kesalahan.

Penyadian RS yang sekarang digunakan sebagai sistem koreksi kesalahan, banyak dijumpai untuk berbagai keperluan, di antaranya [5]:

1. Media penyimpanan data (*Storage devices*), seperti *hard disk, compact disk* (CD), DVD, dan *barcode*
2. Komunikasi nirkabel, seperti telepon bergerak dan *microwave links*
3. Televisi digital
4. Komunikasi satelit, termasuk untuk misi ruang angkasa
5. *Broadband modems* (ADSL, xDSL).

Penyandian RS dengan menyisipkan sandi tambahan ke dalam data aslinya. Data yang telah disandikan bisa jadi disimpan dalam media penyimpanan atau ditransmisikan. Saat data tersebut dibaca kembali atau diterima, kemungkinan telah mengalami kerusakan, misalnya karena goresan pada CD, permukaan *hard disk* yang tidak sempurna, atau interferensi frekuensi radio pada telepon bergerak. Sandi tambahan tersebut memungkinkan pengurai untuk mendeteksi bagian mana dari data yang mengalami kerusakan atau berubah dan kemudian memperbaikinya.

2.5.2.1 Polinomial Generator

Perhitungan pada Reed-Solomon juga menggunakan perhitungan aritmatika *finite field* dari elemen q yang dinotasikan dengan GF(q) dimana jumlah dari

elemen dalam sebuah *finite field* dibentuk dari q = 2^m. Reed-Solomon dengan simbol dari GF(q) memiliki parameter-parameter sebagai berikut[2]:

Panjang sandi blok : n = q-1

Jumlah digit *parity check* : n – k = 2t

Jarak minimum : $d_{\min} = 2t+1$

$$n = 2^m - 1 \tag{2.27}$$

Maka untuk generator polinomial dari sebuah polinomial primitif dengan panjang blok 2^m -1 adalah:

$$g(x) = (x+\alpha)(x+\alpha^2)...(x+\alpha^{2t})$$
$$= g_0 + g_1 x + ... + g_{2t} x^{2t-1} + x^{2t}$$

dimana g(x) memiliki $\alpha, \alpha^2, ..., \alpha^{2t}$ sebagai akar-akar dan koefisien dari GF(2^m).

2.5.2.2 Penyandian

Proses pembentukan kata sandi dengan Reed-Solomon sama dengan pembentukan kata sandi dengan BCH.

Misalkan, $a(x) = a + a_1 x + a_2 x^2 ... + a_{k-1} x^{k-1}$ adalah informasi yang akan dibentuk kata sandinya dan k = n – 2t. Dalam kata sandi dengan bentuk sistematis, digit dari *parity check* (2t) adalah koefisien dari sisa pembagian $h(x) = h_0 + h_1 x + h_2 x^2 ... + h_{2t-1} x^{2t-1}$ yang dihasilkan dari pembagian polinomial informasi $x^{2t} a(x)$ terhadap polinomial g(x).

2.5.2.3 Penguraian

Dalam proses penguraian kata sandi dengan teknik Reed-Solomon diperlukan tiga tahapan yang juga terdapat pada proses penguraian kata sandi dengan BCH, dengan tambahan satu tahapan yaitu perhitungan dari nilai kesalahan. Komponen sindrom 2t didapat dengan menggantikan α^i ke dalam polinomial kata sandi yang diterima r(x) untuk i = 1, 2, ..., 2t. Sehingga:

$$S_1 = r(\alpha)$$
$$S_2 = r(\alpha^2)$$
$$S_{2t} = r(\alpha^{2t}) \tag{2.28}$$

Komponen sindrom Si dapat juga dihitung dengan membagi r(x) dengan $x + \alpha^i$.

$$r(x) = c_i(x)(x + \alpha^i) + b_i \quad (2.29)$$

Dimana sisa pembagian b_i adalah sebuah konstanta dalam GF(2^m). Substitusikan α^i ke dalam persamaan (2.29), maka diperoleh:

$$S_i = b_i$$

Untuk mencari polinomial dari lokasi kesalahan:

$$\tau(x) = (1 + \beta_1 x)(1 + \beta_2 x)...(1 + \beta_v x)$$
$$= 1 + \tau_1 x + ... + \tau_v x$$

dimana $\beta_l = \alpha^{jl}$ untuk l = 1, 2, ..., v adalah sebagai jumlah lokasi kesalahan.

2.5.3 Penggabungan Reed-Solomon (15,5) Dengan BCH (15,5)

Penyandian siklis banyak diimplementasikan pada perangkat keras sehingga teknik-teknik penyandian dan penguraian yang dikembangkan tidak dapat digabungkan karena masing-masing memiliki perangkat keras yang berbeda. Namun jika teknik-teknik ini diimplementasikan pada perangkat DSP prosesor yang menggunakan program simulink untuk rekayasa penggabungannya, maka kendala terhadap perbedaan karakter yang ditemui pada perangkat keras yang terpisah tidak terjadi lagi.

Penggabungan teknik penyandian RS (15,5) dan BCH (15,5) dilakukan secara serial (*concatenated*). Alasan penggabungan kedua teknik secara serial adalah sebagai berikut:

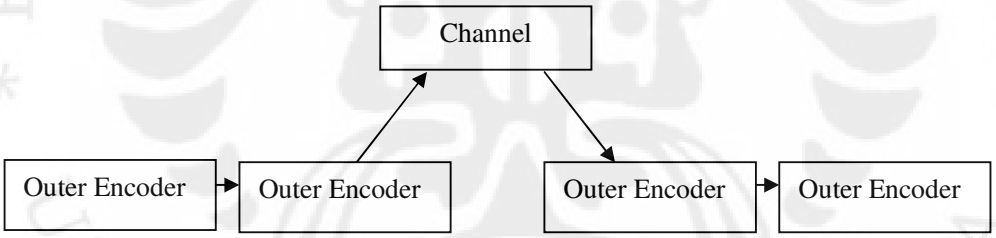
1. Kemudahan dalam melakukan penyandian maupun dalam penguraian. Hal ini disebabkan kecocokan dimensi antara keluaran dari blok penyandi RS (15,5) dan masukan untuk blok penyandi BCH (15,5).
2. Perhitungan yang lebih sederhana jika dibandingkan menggunakan nilai n yang lebih besar.

Pada penggabungan secara serial, proses penyandian akan dilakukan dua tahap dimana tahap pertama yaitu informasi sepanjang 20 bit dengan 4 bit per simbolnya yang akan dikirim disandikan terlebih dahulu dengan teknik RS (15,5). 15 simbol keluaran penyandi RS direpresentasikan oleh 4 bit sehingga panjang kata sandinya menjadi 60 bit. Kemudian pada tahap kedua kata sandi dari RS (15,5) disandikan lagi dengan teknik BCH (15,5) sehingga menghasilkan panjang

kata sandi 180 *bit*. Penyandi dengan teknik RS (15,5) dalam hal ini disebut sebagai *outer encoder* sedangkan penyandi dengan teknik BCH (15,5) disebut sebagai *inner encoder*.

Proses penguraiannya juga melalui dua tahap dimana tahap pertama kata sandi yang diterima diuraikan dengan teknik BCH (15,5). Setelah itu pada tahap kedua kata sandi yang telah diuraikan dengan teknik BCH (15,5) diuraikan lagi dengan teknik RS (15,5). Pengurai BCH (15,5) dalam hal ini disebut sebagai *inner decoder* sedangkan pengurai RS (15,5) disebut sebagai *outer decoder.*

Penggabungan secara serial ini juga efektif untuk mengatasi kesalahan secara acak maupun secara berurutan dimana pola dari *byte* yang tidak dapat dikoreksi dengan *inner code* harus dikorekasi dengan *outer code*. Kesalahan secara acak dapat dikorekasi oleh *inner code* dan untuk beberapa kesalahan secara berurutan yang kemungkinan tidak dapat dikoreksi oleh *inner code* akan dikoreksi oleh *outer code*. Gambar 2.2 menunjukkan tahapan proses *encoding* hingga *decoding*.



Gambar 2.2 Alur Penyandian Dengan Penggabungan 2 Teknik Secara Serial

2.6 *DIGITAL SIGNAL PROCESSING STARTER KIT* (DSK) TMS320C6713[6]

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 dibuat di DSK ini sangat mungkin untuk diterapkan pada DSP C6713 yang sebenarnya.

2.6.1 *Digital Signal Processing 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 opersi matematika dapat dilakukan hanya dalam satu *cycle* dari *clock*. Sehingga DSP mampu melakukan opersi *multiple-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 k*byte* sampai 256 k*byte*.

c. Pengolahan data kontinu

   DSP dapat diaplikasikan pada proses pengolahan yang datanya terus mengalir secara kontinu (*stream*).

d. Konsumsi daya yang rendah

   Mikroprosesor biasa memerlukan daya yang relatif besar. Contohnya dapat dilihat pada *Personal Computer* (PC). Mikroprosesor PC mengkonsumsi daya sekitar 10 sampai 100 watt. Dengan konsumsi daya sebesar ini, jika diaplikasikan pada alat yang menggunakan baterai AA, maka umur baterai mungkin hanya sampai 11 menit. Sedangkan prosesor DSP hanya mengonsumsi daya sekitar 100 miliwatt. Dengan konsumsi daya seperti ini, baterai AA dapat dipakai hingga 18 jam.

e. *Real-time*

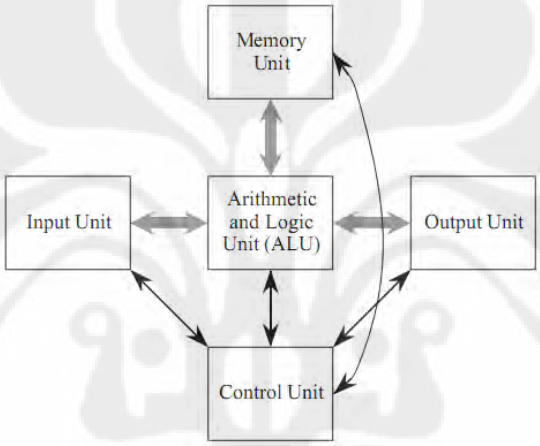
   DSP dapat melakukan pemrosesan sinyal secara *real-time*. Hal ini disebabkan

karena DSP dapat mengolah aliran data yang datang secara serus menerus. *Real-time* juga dapat diartikan tidak adanya jeda waktu (*delay*) antara *input* dan *output*, sehingga saat data dimasukkan ke sebuah sistem, saat itu juga data diproses dan keluar dari sistem.

2.6.2 Arsitektur DSP Prosesor

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

a. Von Neuman

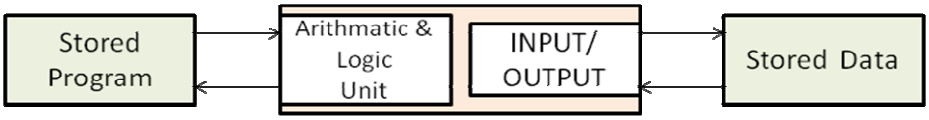


Gambar 2.3 Arsitektur Von Neuman [6]

Pada arsitektur Von Neuman seperti pada Gambar 2.3 terdapat 2 unit operasi dasar, yaitu *Arithmatic and Loogical Unit* (ALU) dan *Input/Output* (I/O). ALU berfungsi untuk melakukan operasi aritmatika dan logika, sedangkan I/O berfungsi untuk mengaturkeluar masuknya data. Ciri arsitektur ini adalah program dan data disimpan pada memori yang sama.

b. Harvard

Unit operasi dasar yang digunakan sama dengan arsitektur Von Neuman. Perbedaannya terletak pada alokasi memorinya. Dapat dilihat pada Gambar 2.4 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 meninggatkan kecepatan proses.



Gambar 2.4 Arsitektur Harvard [6]

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

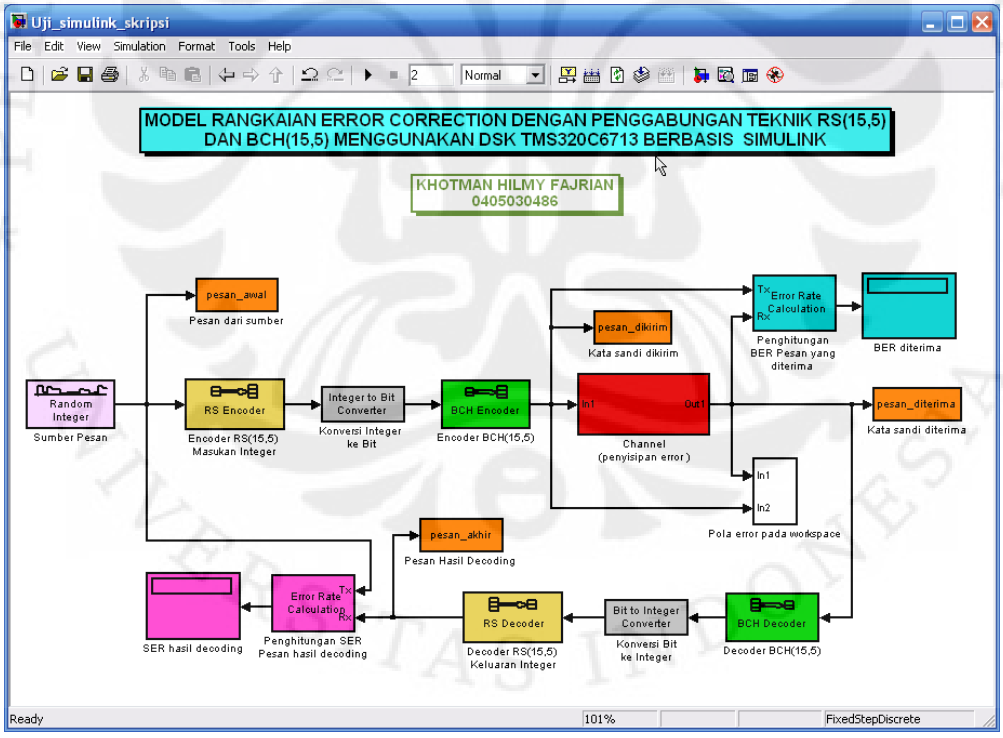
2.6.3 *Starting* DSK TMS320C6713

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 memori*. 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* keUSB pada DSK *board*, akan muncul *Windows Hardware Device Manager*. Kemudian pada saat CCS dimulai, *driver emulation* akan melakukan kontak ke DSK *board.* Lalu CCS akan melakukan proses *download emulation formware* 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 DSKberada dalamkondisi menyala, program baru dapat diluncurkan. USB *controller* tidak dipengaruhi oleh tombol *reset* pada *board* DSK. Saat DSK menyala, USB *controller* mendapatkan daya saat *reset*.

BAB 3
PERANCANGAN SIMULASI

3.1 SIMULINK

Simulink merupakan sebuah perangkat lunak sub-program dari MATLAB yang dirancang untuk digunakan dalam pembuatan model dari suatu sistem terutama di bidang teknik, termasuk di dalamnya sistem pemrosesan sinyal, serta mensimulasikannya sesuai dengan parameter-parameter yang ditentukan. Model sistem dibuat dengan cara merangkai blok-blok yang terdapat di dalam Simulink *library* sesuai dengan fungsi blok yang dikehendaki. Model rangkaian *error correction* dengan penggabungan teknik penyandian Reed-Solomon (15,5) dan BCH (15,5) dibangun dengan merangkaikan blok-blok fungsi yang diperlukan untuk sistem tersebut, sebagaimana dapat dilihat pada Gambar 3.1.

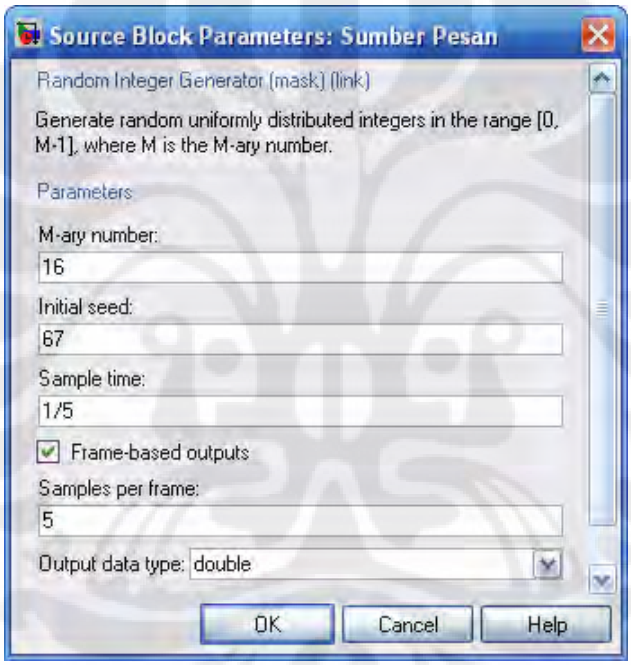


Gambar 3.1 Model Rangkaian *error correction* dengan penggabungan teknik RS (15,5) dan BCH (15,5) pada simulink

Penjelasan tiap blok dari model rangkaian *error correction* pada Gambar 3.1 diuraikan selanjutnya.

3.1.1 Blok "Sumber Pesan"

Blok "Sumber Pesan" adalah sebuah blok yang berfungsi untuk membangkitkan suatu sinyal sebagai pesan atau informasi yang selanjutnya akan disandikan. Blok pembangkit sinyal yang digunakan sebagai sumber adalah blok *Random Integer Generator* yang membangkitkan sinyal berupa bilangan bulat positif berbasis 10 (desimal) secara acak. Parameter dari blok ini disesuaikan dengan karakteristik data masukan dari blok penyandi RS (15,5) yang akan dilalui oleh data tersebut selanjutnya. Tampilan pengaturan parameter blok sumber ini diperlihatkan pada Gambar 3.2.



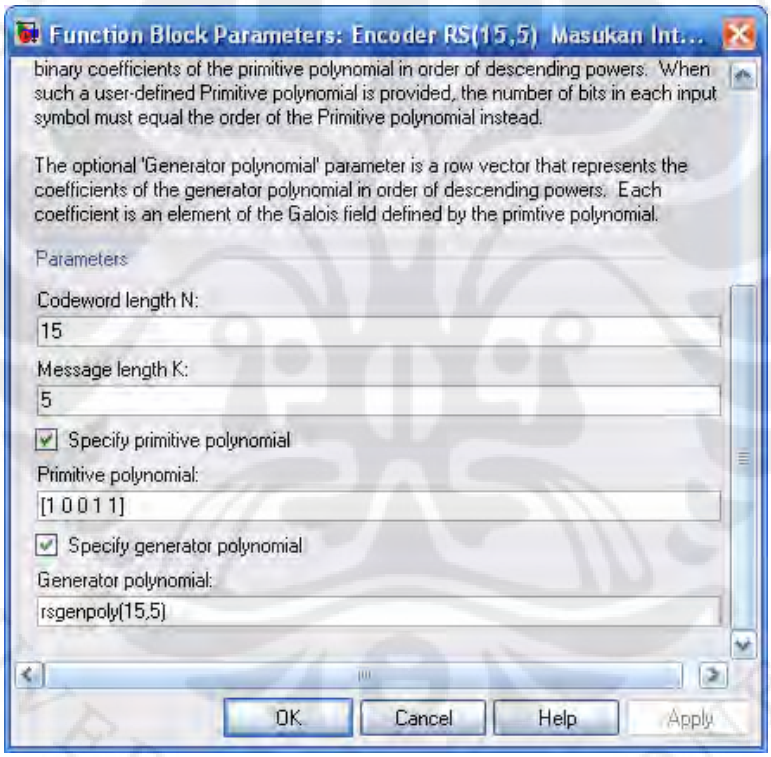
Gambar 3.2 Pengaturan parameter blok "Sumber Pesan"

Parameter 'M-ary number' diberikan nilai 16 yang berarti sinyal yang akan dibangkitkan adalah 16 bilangan pertama dari bilangan cacah, yaitu dari 0 sampai dengan 15. Nilai 16 ini adalah nilai maksimum yang dapat diberikan agar data yang dibangkitkan sesuai dengan karakter blok *encoder* RS (15,5) masukan *integer* yang masukannya harus berupa bilangan bulat positif antara 0 sampai dengan n, dalam hal ini n sama dengan 15. Selanjutnya data yang dibangkitkan dipilih berbetuk *frame-based data* dengan 5 sampel per *frame*-nya sebab blok *encoder* RS (15,5) mempunyai syarat masukannya harus berbentuk *frame-based*

dengan jumlah k sampel per *frame*. Sehingga dengan nilai parameter *sample time* sama dengan 1/5 maka blok ini akan membangkitkan 1 *frame* data yang berisi 5 sampel dengan nilai bulat antara 0 sampai 15 secara acak tiap satu periodenya.

3.1.2 Blok "*Encoder* RS (15,5) Masukan *Integer*"

Blok "*Encoder* RS (15,5) Masukan *Integer*" adalah sebuah blok penyandi Reed-Solomon dengan masukan berupa bilangan bulat positif berbentuk *frame-based data.* Parameter yang ditentukan pada blok ini adalah nilai n dan k serta dapat juga ditentukan polonomial primitifnya dalam bentuk matriks koefisien. Pengaturan parameter blok *Encoder* RS (15,5) ini dapat dilihat pada Gambar 3.3 berikut.



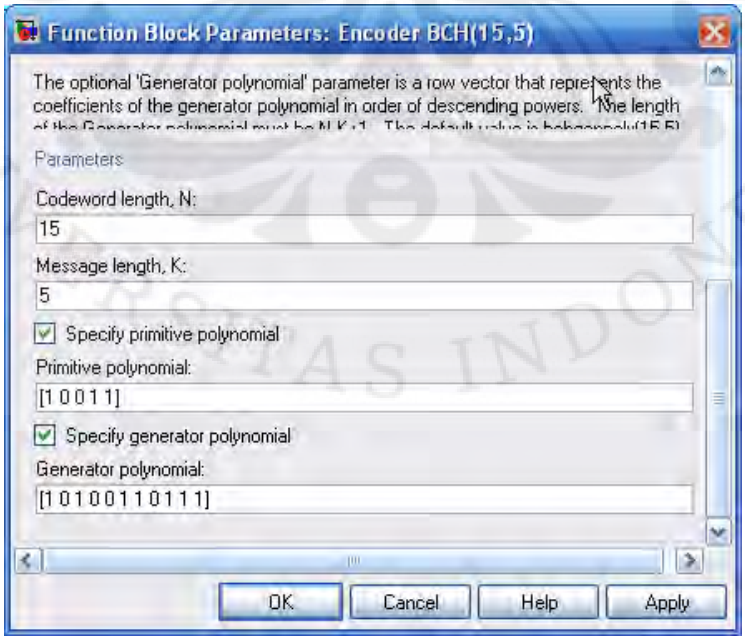
Gambar 3.3 Pengaturan parameter blok *Encoder* RS (15,5)

Nilai 15 pada parameter *Codeword length N* menunjukkan bahwa blok ini akan menghasilkan kata sandi dengan panjang 15 simbol atau kelipatan bulat positifnya. Sedangkan parameter *Message length K* diisikan nilai 5 berarti masukan blok ini berupa data berbentuk *frame* dengan panjang 5 simbol atau kelipatan bulat positifnya. Simbol pada masukan dan keluaran dari blok ini haruslah berupa bilangan bulat yang dapat direpresentasikan oleh 4 digit biner,

yaitu bilangan desimal mulai dari 0 sampai 15. Parameter *Primitive polynomial* diisi dengan vektor baris yang berisi koefisien dari polinomial primitif yang digunakan. Vektor [1 0 0 1 1] mewakili bentuk polinomial berderajat 4 $p(x) = 1 + x + x^4$. Parameter ini boleh juga tidak ditentukan dengan menghilangkan tanda *check* pada *Specify primitive polynomial.* Jika tidak ditentukan nilainya, program simulink sendiri yang mencari nilainya berdasarkan nilai N dan K yang telah ditentukan.

3.1.3 Blok "*Encoder* BCH (15,5)"

Blok "*Encoder* BCH (15,5)" adalah sebuah blok penyandi dengan teknik BCH primitif, yaitu penyandian BCH yang masukannya berupa data biner. Parameter yang harus ditentukan pada blok ini antara lain panjang pesan atau masukan K dan panjang kata sandi N. Nilai yang dimasukkan untuk panjang kata sandi N adalah 15 dan untuk panjang pesan K adalah 5. Sedangkan parameter *Primitive polynomial* dan *Generator polynomial* dapat dituliskan sendiri atau bisa juga tidak, namun hanya ada satu nilai yang sesuai. Dalam skripsi ini penulis menentukan sendiri polinomial primitif dan generatornya. Pengaturan parameter blok *Encoder* BCH (15,5) dapat dilihat pada Gambar 3.4.

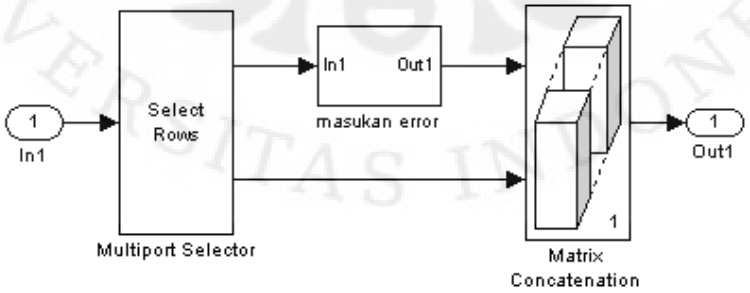


Gambar 3.4 Pengaturan parameter blok *Encoder* BCH (15,5)

Pada Gambar 3.4, nilai [1 0 0 1 1] untuk parameter *primitive polynomial* mewakili polinomial $p(x) = 1 + x + x^4$ dan nilai [1 0 1 0 0 1 1 0 1 1 1] untuk parameter *generator polynomial* mewakili polonomial $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$. Sebelum masuk ke blok *Encoder* BCH (15,5), data yang masih berbentuk *integer* dikonversi terlebih dahulu ke bentuk biner menggunakan blok "Konversi *Integer* ke *Bit*". Blok konversi ini mengubah tiap nilai *integer* yang masuk ke dalam susunan 4 digit biner atau *bit*. Jumlah 4 *bit* ini disesuaikan dengan masukannya yang mempunyai rentang nilai dari 0 sampai dengan 15.

3.1.4 Blok "*Channel* (Penyisipan *Error*)"

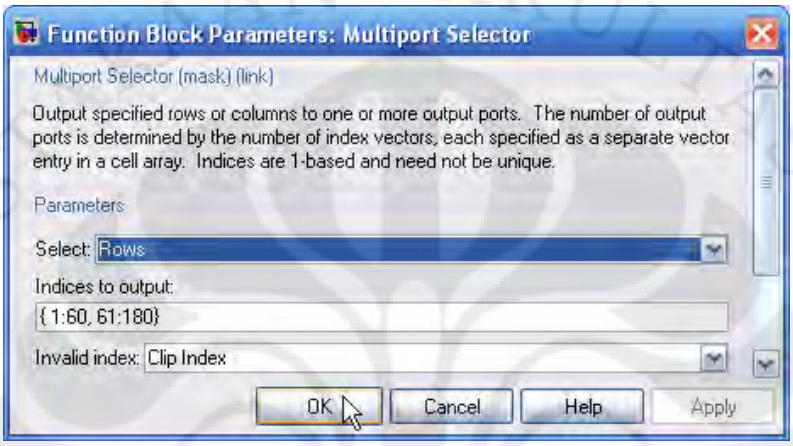
Blok "*Channel* (Penyisipan *Error*) adalah suatu blok subsistem yang berfungsi sebagai penambah kesalahan pada data, yang mewakili saluran transmisi atau media penyimpanan. Dalam uji simulasi yang akan diuraikan pada bab 4, blok ini dibagi menjadi 2 jenis dengan fungsi yang sama namun memberikan pola kesalahan yang berbeda. Kedua pola kesalahan itu adalah pola kesalahan mengelompok dan pola kesalahan menyebar. Penerapan kedua pola kesalahan tersebut ke dalam sistem bertujuan untuk menguji keandalan sistem dalam mengoreksi kesalahan yang mungkin terjadi dengan pola bervariasi. Untuk menghasilkan pola kesalahan mengelompok, blok subsistem "*channel"* ini dibangun dengan rangkaian blok seperti pada Gambar 3.5.



Gambar 3.5 Rangkaian subsistem "C*hannel* (penyisipan *error*)" dengan pola *error* mengelompok

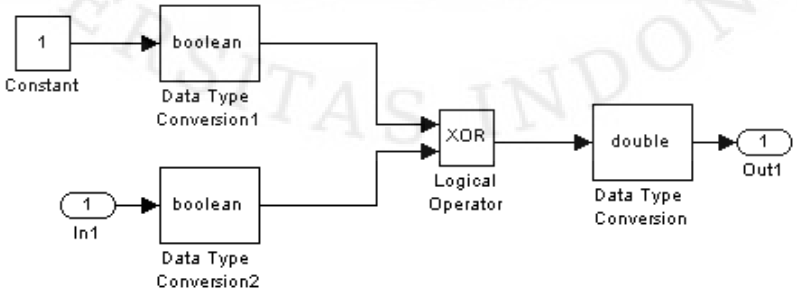
Pada Gambar 3.5, blok "*Multiport Selector"* berfungsi untuk memilih elemen ke berapa dari sebuah *frame* data yang akan disisipkan dengan *error*.

Pemilihan elemen yang akan disisipkan *error* dilakukan dengan cara menentukan parameter pembagian data *frame* yang masuk melalui kotak dialog seperti pada Gambar 3.6. Isian pada parameter '*Indices to output*' menujukkan bahwa tiap *frame* data yang masuk akan dibagi menjadi 2 keluaran dimana keluaran pertama berisi elemen baris ke-1 sampai elemen ke-60 dari data *frame* tersebut. Sedangkan keluaran kedua berisi elemen baris ke-61 sampai elemen ke-180.



Gambar 3.6 Pengaturan parameter blok *Multiport Selector*

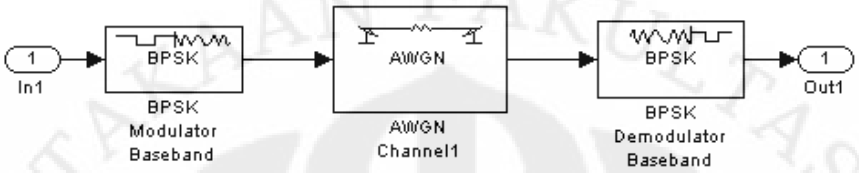
Selain blok *Multiport Selector*, dalam subsistem *channel* juga terdapat subsistem "masukan *error*" yang berfungsi menghasilkan *error* terhadap data yang masuk dengan mengubah nilai data biner 0 menjadi 1 dan 1 menjadi 0. Pengubahan nilai *bit* tersebut dilakukan dengan menambahkannya dengan 1 menggunakan operator logika "XOR", seperti terlihat pada gambar 3.7.



Gambar 3.7 Rangkaian subsistem "masukan *error*"

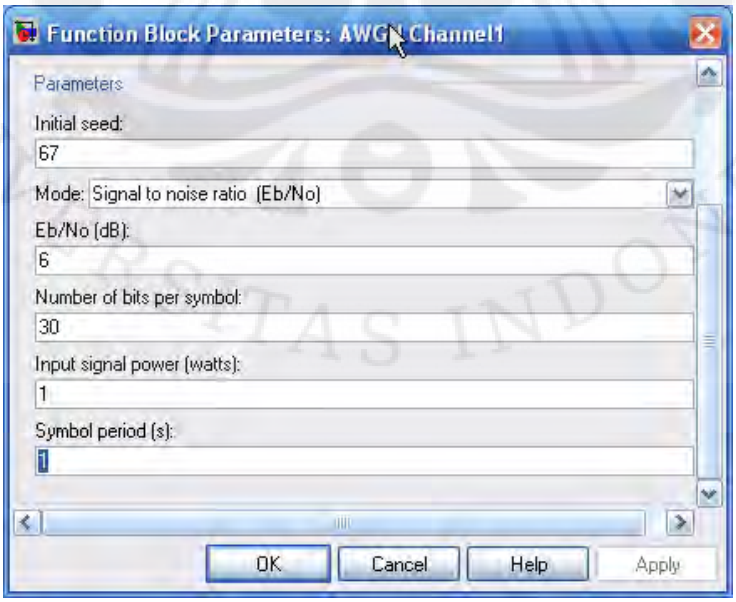
Sedangkan blok "*Matrix Concatenation"* berfungsi menyusun kembali data yang telah terbagi menjadi data yang mengalami *error* dan data yang tidak mengalami *error* ke dalam 1 *frame* dengan urutan elemen yang tidak berubah.

Sedangkan untuk menghasilkan pola kesalahan yang menyebar, subsistem "*channel"* dibangun dengan rangkaian blok seperti pada Gambar 3.8.



Gambar 3.8 Rangkaian subsistem "C*hannel* (penyisipan *error*)" dengan pola menyebar

Rangkaian ini menggunakan modulasi dan demodulasi dengan tipe BPSK (*Binary Phase Shift Keying*) dan dilewatkan pada saluran AWGN (*Additive White Gaussian Noise*). Blok saluran AWGN ini yang berfungsi menambahkan kesalahan pada data secara acak atau menyebar. Parameter yang ada pada blok saluran AWGN di atur sedemikian sehingga menyebabkan kesalahan dengan probabilitas yang dikehendaki. Parameter blok saluran AWGN dapat dilihat pada Gambar 3.9.

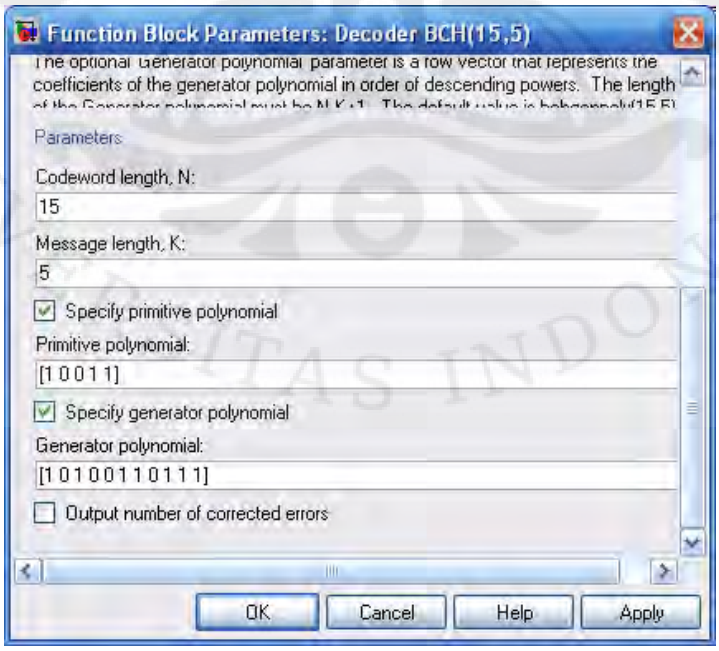


Gambar 3.9 Pengaturan parameter blok saluran AWGN

Untuk parameter Eb/No(dB) dan *number of bits per symbol*, menambahkan nilainya akan menurunkan probabilitas *error* yang ditimbulkan dan mengurangkannya akan menaikkan probabilitas *error*.

3.1.5 Blok "*Decoder* BCH (15,5)"

Blok "*Decoder* BCH (15,5)" merupakan pasangan dari blok "*Encoder* BCH (15,5)" dan berfungsi untuk mendeteksi serta mengoreksi kesalahan yang dialami oleh data selama dalam saluran transmisi. Kemampuannya dalam mengoreksi kesalahan tergantung pada nilai parameter *Codeword length* N dan *Message length* K yang diberikan. Berdasarkan persamaan (2.18) dengan nilai n=15 dan n=5 berarti *encoder* BCH (15,5) mampu mengoreksi 3 *bit error* pada tiap 15 *bit* kata sandi. Blok ini menerima data masukan *frame* sepanjang 180 *bit* dan menghasilkan keluaran dengan panjang 60 *bit*. Data biner ini kemudian diubah ke *integer* menggunakan blok "Konversi *Bit* ke *Integer*" , dimana tiap 4 *bit* dikonversi menjadi sebuah simbol berupa bilangan integer bernilai 0 hingga 15. Sehingga panjang data 60 *bit* dikonversi menjadi 15 simbol *integer*. Pengubahan ini diperlukan untuk proses *decoding* selanjutnya oleh *decoder* RS. Gambar 3.10 memperlihatkan parameter-parameter untuk blok *decoder* BCH (15,5).



Gambar 3.10 Pengaturan parameter blok "*Decoder* BCH (15,5)

3.1.6 Blok "*Decoder* RS (15,5) Keluaran *Integer*"

Blok "*Decoder* RS (15,5) Keluaran *Integer*" adalah pasangan dari blok "*Encoder* RS (15,5) Masukan *Integer*" dan berfungsi untuk mendeteksi serta mengoreksi kesalahan yang dialami oleh data selama berada dalam saluran transmisi. Kemampuannya dalam mengoreksi kesalahan juga tergantung pada nilai N dan K yang diberikan sebagaimana blok *encoder* BCH. Berdasarkan persamaan n – k = 2t, dengan nilai n =N=15 dan k=K=5 menunjukkan bahwa *decoder* RS (15,5) mampu mengoreksi maksimum 5 simbol *error*. Masukan blok ini adalah data *frame* sepanjang 15 simbol *integer*, yaitu keluaran blok *decoder* BCH yang telah dikonversi dari *bit* ke *integer.* Sedangkan keluarannya adalah data *frame* sepanjang 5 simbol *integer* dan merupakan hasil akhir dari sistem *error correction* yang dirancang. Idealnya hasil akhir ini diharapkan sama dengan data awal yang dibangkitkan oleh blok "*Random Integer*".

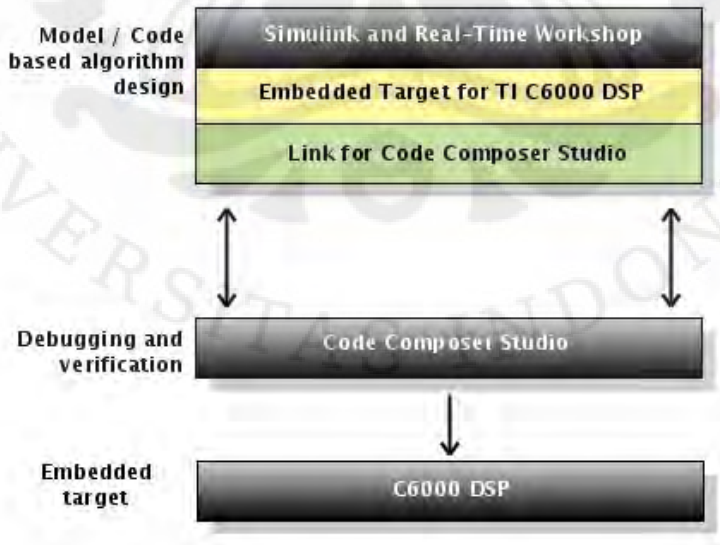
3.1.7 Blok "Perhitungan BER" dan "Perhitungan SER"

Blok "Perhitungan BER" berfungsi melakukan perhitungan BER (*Bit Error* Rate) atau laju terjadinya kesalahan *bit* dengan cara membandingkan jumlah *bit error* yang masuk terhadap jumlah keseluruhan *bit* yang masuk. Data yang dihitung BER-nya adalah data yang diterima dibandingkan dengan data yang dikirim. Tujuan penghitungan BER ini adalah untuk mengetahui tingkat kesalahan yang ditimbulkan oleh saluran yang digunakan dalam proses transmisi.

Blok "Perhitungan SER" berfungsi melakukan perhitungan SER (*Symbol Error Rate*) atau laju terjadinya kesalahan simbol pada pesan akhir hasil keseluruhan proses *decoding*. Perhitungannya dilakukan dengan membandingkan jumlah simbol yang mengalami kesalahan dari data atau pesan akhir terhadap jumlah seluruh simbol yang telah diproses dari pesan awal yang dibangkitkan oleh *interger generator.* Tujuan penghitungan SER ini adalah untuk mengetahui keandalan sistem *error correction* yang dibuat dalam menanggulangi kedua pola kesalahan yang mungkin terjadi. Dengan membandingkan nilai SER terhadap nilai BER, dapat diketahui secara kasar seberapa besar sistem tersebut dapat mereduksi kesalahan yang terjadi dalam saluran transmisi atau media penyimpanan.

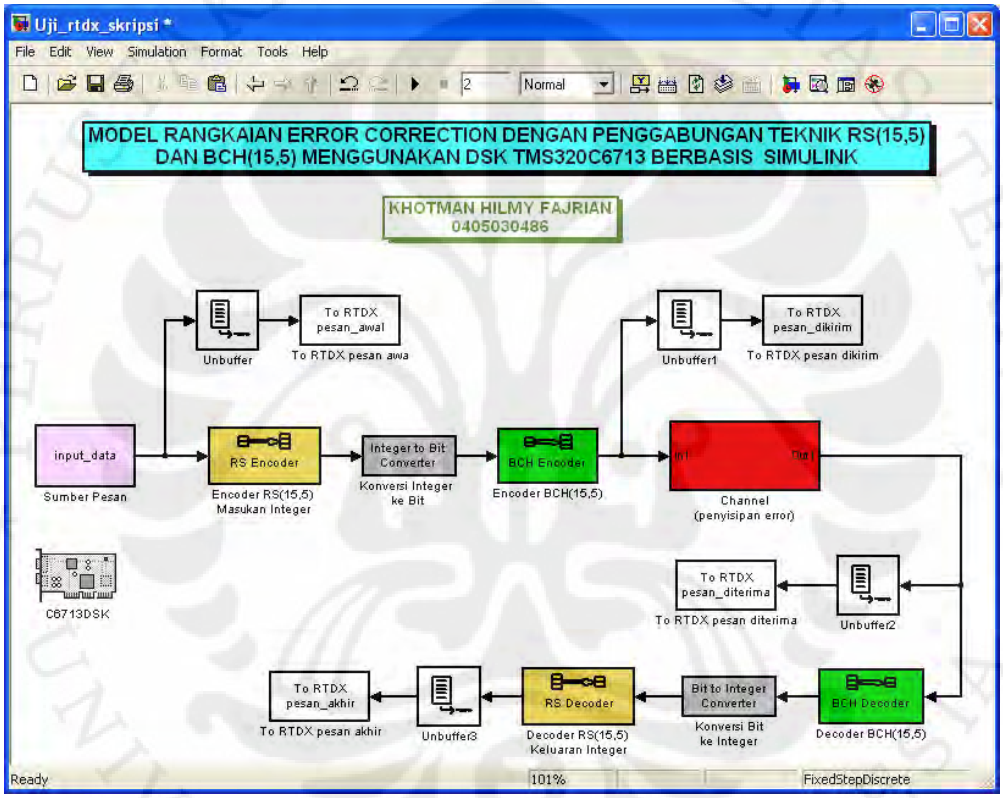
3.2 PENERAPAN MODEL SIMULINK KE DSK TMS320C6713

Model rangkaian *error correction* yang telah dibuat pada simulink dapat kemudian dimasukkan dalam DSK TMS320C6713 dengan hanya melakukan beberapa setting pada *Configuration Parameters*. Proses *debugging* dari SIMULINK® ke DSK memerlukan perangkat lunak lain yaitu Code Composer Studio (CCS) yang merupakan sebuah komposer bahasa C/C++ yang digunakan untuk memrogram papan DSK dengan menggunakan bahasa C/C++. Maka sebenarnya yang dilakukan oleh SIMULINK® adalah membangun kode pemrograman dalam bahasa C/C++ dari model yang disimulasikan. Kode pemrograman tersebut kemudian dijalankan oleh CCS. Dan kemudian SIMULINK® dan papan DSK TMS320C6713 dapat berkomunikasi melalui kode-kode pemrograman yang dibuat oleh SIMULINK®. Inilah kemudahan yang diberikan oleh SIMULINK®, yaitu kita tidak perlu membuat dan membangun kode pemrograman dari model yang kita simulasikan. Kita hanya cukup membangun blok model pada SIMULINK® dan dengan mudah kita akan mendapatkan kode pemrograman dalam bahasa C/C++ yang dapat digunakan untuk membuat rancang bangun dari model yang telah kita buat dalam SIMULINK® pada DSK TMS320C6713. Gambar 3.11 menunjukkan algoritma dari penerapan SIMULINK pada DSK.



Gambar 3.11 Algoritma penanaman model simulink ke DSK TMS320C6713 sebagai *embedded target* [7]

Perancangan simulasi rangkaian *error correction* dengan penggabungan teknik RS (15,5) dan BCH (15,5) pada DSK TMS320C6713 dilakukan tetap menggunakan model simulink yang telah dibuat dengan beberapa perubahan yaitu pada blok sumber dan blok pengambil data *to workspace* serta penambahan blok *target preference* C6713DSK yang tersedia pada *library* simulink. Gambar 3.12 menunjukkan perubahan model rangkaian simulink menjadi model yang akan diterapkan pada DSK.



Gambar 3.12 Model rangkaian *error correction* dengan penggabungan teknik RS (15,5 dan BCH (15,5) untuk DSK TMS320C6713

Pada Gambar 3.12 terlihat bahwa blok "Sumber Pesan" yang semula menggunakan blok "*random* integer" diganti dengan blok "*from workspace*" dengan nama variabel "input_data". Penggunaan blok ini bertujuan agar dapat ditentukan sendiri pesan yang akan diproses saat pengujian menggunakan DSK TMS320C6713. Kemudian, blok *to workspace* yang semula digunakan untuk mengambil data dari blok yang diinginkan ke dala *workspace* Matlab diganti dengan blok "To RTDX". RTDX adalah suatu fasilitas yang disediakan oleh

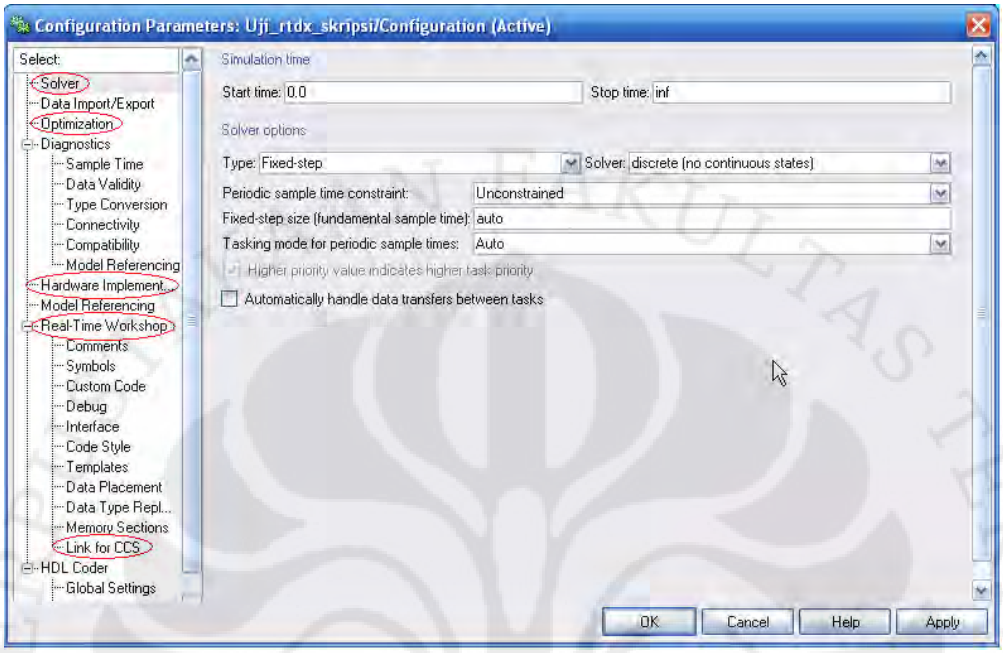
Texas Instrument untuk melakukan pertukaran data antara *target* dan *host. Target* dalam hal ini adalah DSK TMS320C6713, sedangkan *host* adalah prorgam Matlab yang ada di PC atau laptop. Blok "*To* RTDX" yang disediakan simulink berfungsi untuk mengambil data dari DSK yang terhubung ke blok tersebut. Sedangkan untuk mengamati keluaran dari papan DSK menggunakan *oscilloscp,* blok DAC perlu ditambahkan ke bagian yang ingin diamati sebagai keluaran.

Selain itu juga dibutuhkan *Real Time Workshop, Embeded Target for TI (Texas Instrumen) C6000 DSP*, dan *Link for CCS* untuk menghubungkan simulink dengan DSK. Komponen-komponen yang diperlukan tersebut ditemukan pada simulink dan harus dilakukan pengaturan konfigurasi parameter. Pengaturan konfigurasi parameter model rangkaian *error correction* untuk ketiga hal tersebut dailakukan dengan memilih menu *simulation* yang ada pada *toolbar* kemudian pilih konfigurasi parameter. Kotak dialog konfigurasi parameter terlihat pada Gambar 3.13. Pengaturan yang perlu dilakukan antara lain [6]:

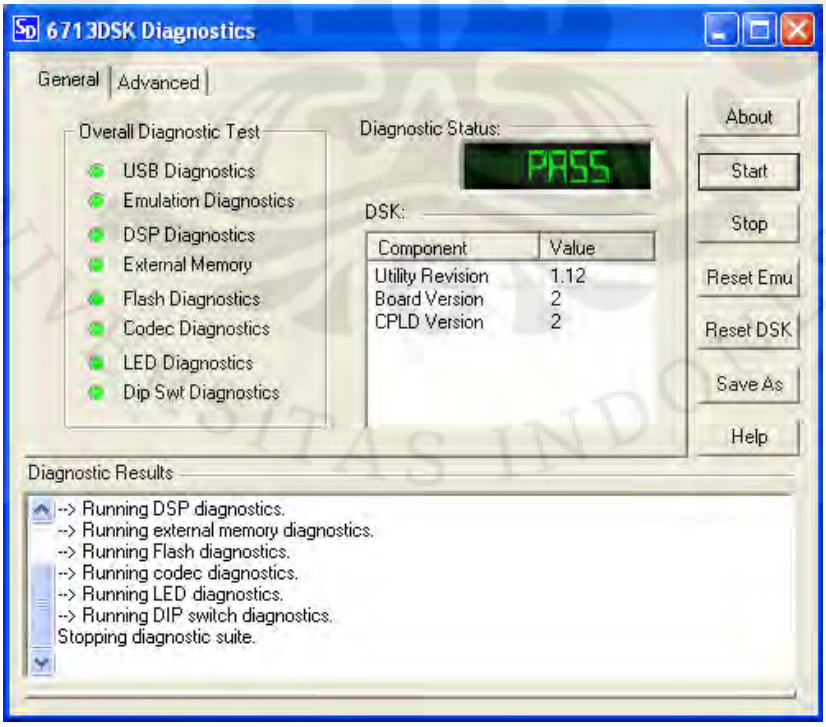
1. Pada *tab solver,* mengubah *type* menjadi *fixed* dan *solver* menjadi *discrete.*
2. Pada *tab optimization,* melakukan *uncheck block reduction* dan *implement logic signal as boolean data* yang terdapat pada menu *simulation and code generation.*
3. Pada *tab hardware implementation,* mengubah *device type* yang terdapat pada menu *embeded 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.

Setelah model rangkaian *error correction* disesuaikan, pemeriksaan koneksi perangkat DSK ke komputer harus dilakukan. DSK harus terhubung ke komputer menggunakan kabel USB dan untuk memastikan keduanya terhubung dengan benar maka proses diagnosa perlu dilakukan. Caranya adalah dengan menggunakan program 6713 *diagnostic* yang disediakan oleh CCS. Jika DSK

terhubung dengan baik ke komputer, maka di kotak dialog program *diagnostic* tersebut akan muncul pesan 'PASS' seperti pada Gambar 3.14.



Gambar 3.13 Konfigurasi Parameter Model



Gambar 3.14 Proses *Diagnostic*

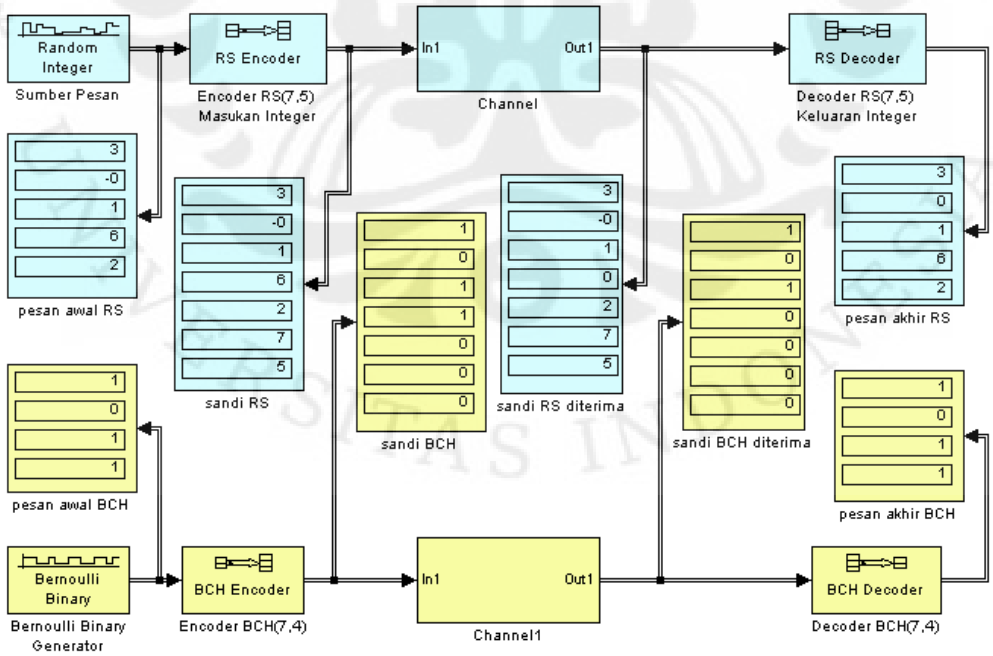
Kemudian dilakukan proses *Incremental Building* yang dapat dilakukan dengan memilih menu yang terdapat di *tools* lalu *real time workshop* kemudian pilih *build model*. Setelah dilakukan proses *build model,* maka matlab akan membuat kode program dalam bahasa C sebagai pengganti blok simulink untuk dijalankan di DSK.

BAB 4
PENGUJIAN DAN ANALISIS SISTEM

Pengujian terhadap model rangkaian *error correction* dilakukan dengan dua metode, yaitu pengujian simulasi menggunakan simulink dan pengujian dengan perangkat DSK TMS320C6713. Kedua metode pengujian tersebut menggunakan media komputer sebagai penampil akhirnya. Parameter yang akan dilihat dari pengujian adalah nilai kesalahan atau *bit error rate* (BER) dari data yang diterima terhadap data yang dikirim serta data akhir hasil *decoding* terhadap data yang dibangkitkan oleh blok sumber.

4.1 PERBANDINGAN HASIL UJI SIMULINK TERHADAP HASIL PERHITUNGAN MANUAL

Untuk membandingkan hasil uji penyandian menggunakan Simulink terhadap hasil perhitungan secara manual berdasarkan teori, maka diambil data hasil pemrosesan sinyal menggunakan teknik penyandian RS (7,5) dan BCH (7,4) secara terpisah menggunakan model Simulink seperti pada Gambar 4.1.



Gambar 4.1 Model Simulink penyandian RS (7,5) dan BCH (7,4)

Pemilihan parameter tersebut hanya untuk mendapatkan proses yang paling sederhana untuk perhitungan secara manual. Hasil pengujian menggunakan Simulink untuk kedua penyandian tersebut dapat dilihat pada Tabel 4.1. Tujuan pembandingan ini adalah untuk membuktikan apakah proses penyandian oleh Simulink sesuai dengan teori.

Tabel 4.1 Data hasil pengujian Simulink untuk penyandian RS (7,5) dan BCH (7,4)

Teknik Penyandian	Pesan	Kata Sandi	Kata Sandi dengan Kesalahan	Hasil Decoding
RS	[2 6 1 0 3]	[5 7 2 6 1 0 3]	[5 7 2 0 1 0 3]	[2 6 1 0 3]
BCH	[1 1 0 1]	[0 0 0 1 1 0 1]	[0 0 0 0 1 0 1]	[1 1 0 1]

Kemudian dilakukan perhitungan secara manual menggunakan persamaan-persamaan yang terdapat dalam teori penyandian RS (7,5) dan BCH (7,4) dengan operasi aritmatik terhadap koefisien α mengacu pada tabel elemen GF (2^3) yang terdapat pada lampiran. Perhitungannya diuraikan sebagai berikut.

4.1.1 Penyandian RS

- Pesan **m** = [2 6 1 0 3] = [010 110 001 000 011]
- Polinomial pesan $m(x) = \alpha + \alpha^3 x + \alpha^2 x^2 + \alpha^4 x^4$
- Polinomial generator $g(x) = (x+\alpha)(x+\alpha^2) = \alpha^3 + \alpha^4 x + x^2$
- $x^{n-k} m(x) = x^{7-5}(\alpha + \alpha^3 x + \alpha^2 x^2 + \alpha^4 x^4) = \alpha x^2 + \alpha^3 x^3 + \alpha^2 x^4 + \alpha^4 x^6$
- $\frac{x^{n-k} m(x)}{g(x)} = v(x) + \frac{h(x)}{g(x)} \rightarrow$

$$\frac{\alpha x^2 + \alpha^3 x^3 + \alpha^2 x^4 + \alpha^4 x^6}{\alpha^3 + \alpha^4 x + x^2} = \alpha^3 + \alpha x + \alpha x^2 + \alpha x^3 + \alpha^4 x^4 + \frac{\alpha^6 + \alpha^5 x}{\alpha^3 + \alpha^4 x + x^2}$$

- Polinomial pariti $h(x) = \alpha^6 + \alpha^5 x$
- Polinomial kata sandi

$$c(x) = h(x) + x^{n-k} m(x) = \alpha^6 + \alpha^5 x + \alpha x^2 + \alpha^3 x^3 + \alpha^2 x^4 + \alpha^4 x^6$$

→ **c** = [101 111 010 110 001 000 011] = [5 7 2 6 1 0 3]

- Kata sandi yang diterima **r** = [5 7 2 0 1 0 3] = [101 111 010 000 001 000 011]

→ $r(x) = c(x) + e(x) = \alpha^6 + \alpha^5 x + \alpha x^2 + \alpha^2 x^4 + \alpha^4 x^6$

- Sindrom $S_i = r(\alpha^i)$ → $S_1 = \alpha^6 + \alpha^6 + \alpha^3 + \alpha^6 + \alpha^{10} = \alpha^6$

$$S_1 = \alpha^6 + \alpha^7 + \alpha^5 + \alpha^{10} + \alpha^9 = \alpha^2$$

- Letak kesalahan σ → $[S_1][\sigma] = [S_2]$ → $[\sigma] = [S_1]^{-1}[S_2]$

→ $[\sigma] = [\alpha^6]^{-1}[\alpha^2] = \frac{1}{\alpha^6}\alpha^2 = \frac{\alpha^7}{\alpha^6}\alpha^2 = \alpha^3$

Sehingga kesalahan terletak pada posisi α^3

- Nilai kesalahan *e* → $[\sigma][e] = [S_1]$ → $[e] = [\sigma]^{-1}[S_1]$

→ $[e] = [\alpha^3]^{-1}[\alpha^6] = \frac{1}{\alpha^3}\alpha^6 = \frac{\alpha^7}{\alpha^3}\alpha^6 = \alpha^{10} = \alpha^3$

Kesalahan bernilai α^3 di posisi α^3, sehingga

$$e(x) = \alpha^3 x^3$$

- Kata sandi hasil *decoding* $c'(x) = r(x) + e(x)$

$$c'(x) = \alpha^6 + \alpha^5 x + \alpha x^2 + \alpha^2 x^4 + \alpha^4 x^6 + \alpha^3 x^3$$

$$= \alpha^6 + \alpha^5 x + \alpha x^2 + \alpha^3 x^3 + \alpha^2 x^4 + \alpha^4 x^6$$

→ **c'** = [101 111 010 110 001 000 011]

= [5 7 2 6 1 0 3]

- Pesan hasil *decoding* **m' =** [2 6 1 0 3]

4.1.2 Penyandian BCH

- Pesan **m**= [1 1 0 1] → $m(x) = 1+x+x^3$
- Polinomial generator $g(x) = p(x) = 1 + x + x^3$
- $x^{n-k}m(x) = x^{7-4}(1 + x + x^3) = x^3 + x^4 + x^6$
- $\frac{x^{n-k}m(x)}{g(x)} = v(x) + \frac{h(x)}{g(x)}$ → $\frac{x^3 + x^4 + x^6}{1 + x + x^3} = x^3 + 0$
- Polinomial pariti $h(x) = 0$ → **h** = [0 0 0]
- Polinomial kata sandi $c(x) = h(x) + x^{n-k}m(x) = x^3 + x^4 + x^6$

→ **c =** [0 0 0 1 1 0 1]

- Kata sandi yang diterima **r =** [0 0 0 0 1 0 1]

→ $r(x) = c(x) + e(x) = x^4 + x^6$

- $\frac{r(x)}{g(x)} = v(x) + \frac{s(x)}{g(x)}$ → $\frac{x^4 + x^6}{1 + x + x^3} = x^3 + \frac{x^3}{1 + x + x^3}$
- Polinomial sindrom $s(x) = x^3$

- $c'(x) = s(x) + r(x) = x^3 + x^4 + x^6$ → **c'** = [0 0 0 1 1 0 1]
- Pesan hasil *decoding* **m'** = [1 1 0 1]

Perhitungan secara manual berdasarkan teori penyandian RS dan BCH di atas memberikan hasil yang sama dengan hasil pemrosesan menggunakan model Simulink, baik hasil *encoding* maupun hasil *decoding*. Hal ini menunjukkan bahwa proses yang dilakukan oleh model Simulink sesuai dengan teori penyandian RS dan BCH.

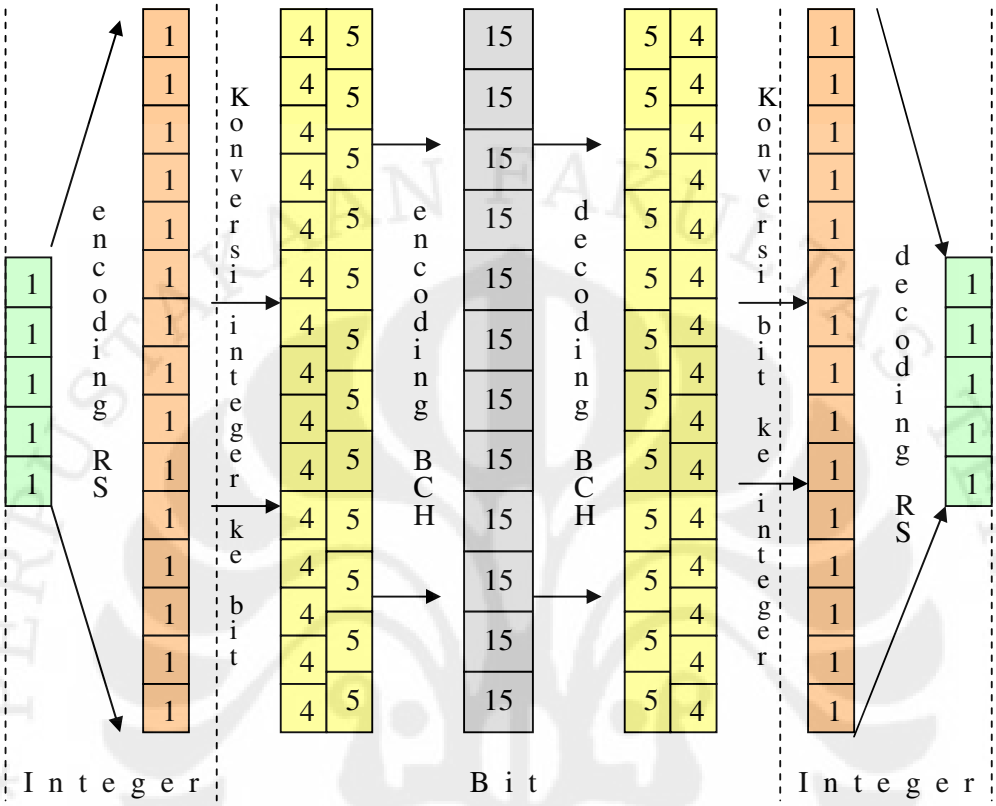
4.2 PENGUJIAN DENGAN SIMULINK

Sebagaimana diuraikan pada bab 3, pengujian sistem dengan program simulink menggunakan model rangkaian *error correction* pada Gambar 3.1. Pengujian dilakukan dengan cara memuat data yang diproses per *frame* ke *workspace* Matlab yang meliputi pesan awal, pesan dikirim, pesan diterima, pesan akhir, dan pola kesalahan. Variasi parameter yang diberikan saat pengujian berupa perbedaan pola kesalahan, yaitu pola kesalahan mengelompok dan menyebar. Pola kesalahan menyebar juga dibagi lagi menjadi 2 jenis, yaitu kesalahan menyebar yang benar-benar merata dan kesalahan menyebar yang acak.

4.2.1 Pola Kesalahan Mengelompok

Parameter yang diperhatikan dalam pengujian ini adalah tingkat kesalahan yang diberikan pada saluran dibuat mendekati kemampuan teknik penyandian RS (15,5) dalam mendeteksi dan mengoreksi kesalahan yang terjadi, yaitu dengan nilai BER sebesar 0,3333 atau sekitar 60 *bit* dari 180 *bit* per *frame*. Pemilihan tingkat kesalahan hanya berdasarkan kemampuan teknik penyandian RS (15,5) karena penyandian ini memiliki kemampuan yang baik untuk mengatasi kesalahan dengan pola mengelompok. Untuk mengetahui berapa jumlah *bit error* maksimum yang dapat dikoreksi, dilakukan pergeseran posisi kelompok *bit* yang mengalami kesalahan. Pengujian diawali tanpa menyisipkan *error* dan dipasangkan blok *display* untuk menampilkan data dari tiap tahap penyandian. Tujuannya adalah untuk mengetahui pola atau skema pemrosesan data pada tiap tahap penyandian.

Dari hasil pengujian awal tersebut, didapat suatu pola pemrosesan data seperti pada Gambar 4.2.

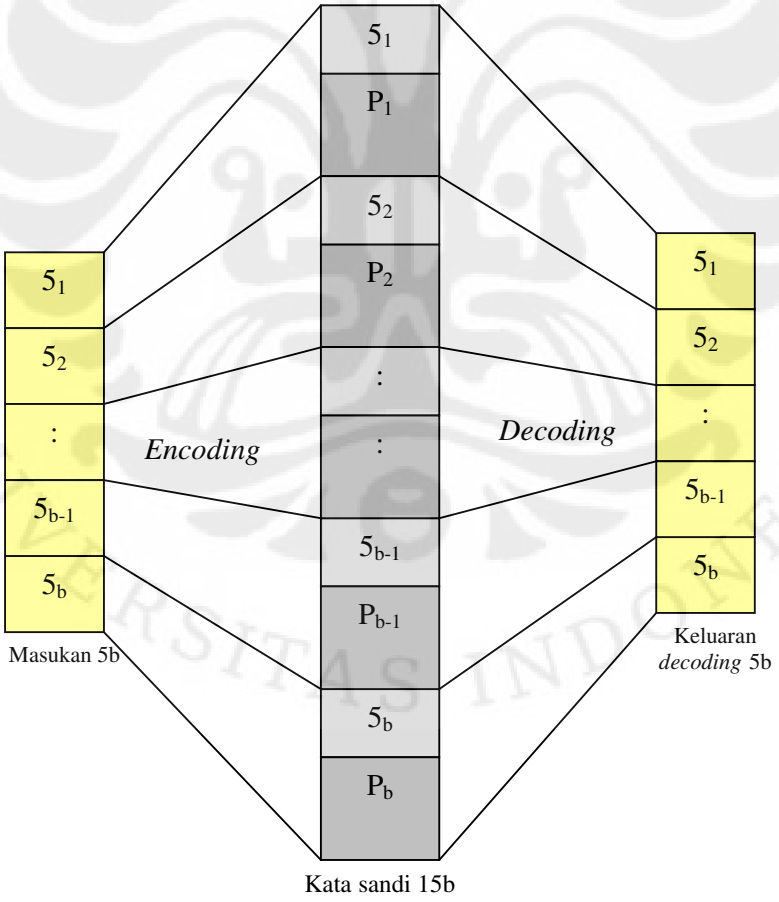


Gambar 4.2 Skema pemrosesan 1 *frame* data dengan penggabungan RS (15,5) dan BCH (15,5)

Tahap pemrosesan data per *frame* dari kiri ke kanan pada Gambar 4.2 dapat diuraikan sebagai berikut:

Sebuah pesan asli dibangkitkan oleh *integer random generator* berupa data *frame* sepanjang 5 simbol *integer*. Pesan asli yang belum diproses ini diwakili oleh blok berwarna hijau di sebelah kiri. Angka-angka yang tercantum di dalam kotak menujukkan jumlah simbol untuk *integer* dan *bit* untuk biner dan bukan merupakan nilai data di dalamnya. Selanjutnya, sinyal asli sepanjang 5 simbol tersebut mengalami proses *encoding* dengan teknik RS (15,5). Proses *encoding* ini menghasilkan sebuah kata sandi sepanjang 15 simbol yang diwakili oleh blok berwarna coklat di sebelah kiri.

Kemudian kata sandi RS (15,5) sepanjang 15 simbol dikonversi ke biner dengan 4 *bit* untuk tiap simbolnya sehingga menghasilkan data *frame* sepanjang 4 *bit*/simbol x 15simbol sama dengan 60 *bit*. Hasil konversi tersebut diwakili oleh blok berwarna kuning di sebelah kiri. Blok ini memiliki dua sisi. Sisi kiri segmentasinya berisi 4 *bit* yang berarti 4 *bit* per simbol yang dikonversi. Sedangkan sisi kanan segmentasinya berisi 5 *bit*, menujukkan pembagian 5 *bit* per *frame* untuk proses *encoding* selanjutnya dengan teknik BCH (15,5). Blok *encoder* BCH (15,5) pada dasarnya menyandikan 5 *bit* masukan menjadi 15 *bit* kata sandi. Selain itu dapat juga memroses 5b *bit* masukan menjadi 15b kata sandi dengan b = 1, 2, 3, 4, ..., namun tetap dengan proses dasarnya yaitu menyandikan tiap 5 *bit* masukan menjadi 15 kata sandi. Proses ini diilustrasikan pada Gambar 4.3 berikut.



Gambar 4.3 Ilustrasi penyandian BCH (15,5) dengan panjang masukan 5b

Gambar 4.2 menunjukkan karakteristik blok *encoder* BCH (15,5) dan *decoder* BCH (15,5) dalam simulink jika diberi masukan dengan panjang kelipatan dari 5, dalam kasus ini masukannya 60 atau 12 x 5 *bit*. Proses *encoding* tidak sekaligus dilakukan terhadap 60 *bit* masukan dan menghasilkan kata sandi180 *bit*, melainkan dari 60 *bit* tersebut disandikan per 5 *bit* dari 5 *bit* pertama sampai 5 *bit* ke-12 dan menghasilkan 15 *bit* kata sandi pertama sampai 15 *bit* kata sandi ke-12. Demikian juga untuk proses *decoding*, namun dengan proses yang berkebalikan dengan *encoding*. Konsekuensi dari karakteristik tersebut adalah *decoder* BCH (15,5) tidak dapat mengoreksi kesalahan sebanyak 20 persen atau sekitar 36 *bit* yang menumpuk dari 180 *bit* kata sandi yang di-*decode*. Hal ini disebabkan karena *decoder* hanya mampu mengoreksi kesalahan 20 persen atau 3 *bit* dari tiap 15 *bit* kata sandi. Dengan kata lain, *decoder* BCH (15,5) dengan masukan 180 *bit* mampu mengoreksi 20 persen kesalahan jika kesalahan itu tersebar secara merata sehingga tidak ada kesalahan yang lebih dari 3 *bit* dalam tiap segmen 15 *bit.* Jumlah 3 *bit* ini adalah jumlah yang pasti dapat dikoreksi di manapun letak kesalahannya. Karakteristik lain yang perlu dipertimbangkan dari proses *decoding* BCH (15,5) yang diperoleh dari pengujian dengan simulink untuk kesalahan lebih dari 3 *bit* yang mengelompok dapat dilihat pada Tabel 4.2 dan Tabel 4.3 berikut.

Tabel 4.2 Karakteristik *decoding* BCH (15,5) dengan *error* lebih dari 3 *bit* mengelompok di belakang

Jumlah *bit error* mengelompok di belakang	Jumlah *bit* salah setelah *decoding*	Letak *bit error* setelah *decoding* (*bit* ke-)
4	0	-
5	0	-
6	1	5
7	0	-
8	0	-
9	1	3
10	0	-
11	1	5
12	5	semua

Tabel 4.3 Karakteristik *decoding* BCH (15,5) dengan *error* lebih dari 3 *bit* mengelompok di depan

Jumlah *bit error* mengelompok di depan	Jumlah *bit* benar setelah *decoding*	Letak *bit* benar setelah *decoding* (*bit* ke-)
4	1	5
5	0	-
6	1	3
7	0	-
8	0	-
9	1	5
10	0	-
11	0	-
12	0	-

Sedangkan karakteristik yang perlu dipertimbangkan dari proses *decoding* RS (15,5) yang diperoleh dari pengujian dengan simulink untuk kesalahan lebih dari 5 simbol yang mengelompok di belakang dapat dilihat pada Tabel 4.4. Untuk kesalahan lebih dari 5 simbol yang mengelompok di depan, hasil *decoding* RS (15,5) masih menyisakan *error* 100 persen.

Tabel 4.4 Karakteristik *decoding* RS (15,5) dengan *error* lebih dari 5 simbol mengelompok di belakang

Jumlah simbol *error* mengelompok di belakang	**Jumlah simbol *error* setelah *decoding***
6	0
7	0
8	0
9	0
10	5

Tabel 4.4 menunjukkan bahwa *decoder* RS (15,5) mampu mengoreksi hingga 9 simbol *error* yang mengelompok di belakang atau akhir dari sebuah data *frame,* meskipun pada dasarnya hanya mampu mengoreksi maksimum 5 simbol *error*. Hal ini dimungkinkan karena 9 simbol *error* tersebut mengelompok di belakang, atau dengan kata lain dialami oleh simbol-simbol tambahan dan bukan

simbol-simbol pesan. Sehingga meskipun pada proses koreksi *error* dalam tahapan *decoding* masih menyisakan *error,* simbol *error* tersebut akan hilang seiring disisihkannya simbol-simbol tambahan. Berdasarkan Tabel 4.4, sistem dapat mengoreksi kesalahan hingga 116 *bit* yang mengelompok di belakang atau mulai *bit* ke-65 sampai *bit* ke-180.

Kemudian dilakukan pengujian dengan variasi jumlah *bit error* mengelompok dan letaknya untuk mengetahui tingkat koreksi terhadap kesalahan yang mengelompok mulai dari *bit* sebelum *bit* ke-65. Berdasarkan karakteristik penyandian BCH seperti pada Gambar 4.3, Tabel 4.2 dan Tabel 4.3, skema pemrosesan data *frame* pada Gambar 4.2 diperoleh hasil untuk kesalahan mengelompok dengan letak seperti pada Tabel 4.5.

Tabel 4.5 Hasil pengujian sistem menggunakan simulink dengan *error* mengelompok sebelum *bit* ke-65

Jumlah *bit error* mengelompok yang dapat dikoreksi	**Letak *bit error* (keadaan khusus)**
65	Mulai *bit* ke-5, 51, 53, 54, 56 dan seterusnya
64	Mulai *bit* ke-6, 51, 53, 54, 56 dan seterusnya
63	Mulai *bit* ke-1, 51, 53, 54, 56 dan seterusnya
62	Mulai *bit* ke-1 sampai ke-2, 8, 51, 53, 54, 56 dan seterusnya
61	Mulai *bit* ke-1 sampai ke-3, 9, 51, 53, 54, 56 dan seterusnya
60	Mulai *bit* ke-1 sampai ke-5, 10, 51, 53, 54, 56 dan seterusnya
59	Mulai *bit* ke-1 sampai ke-6, 11, 20, 21, 35, 51, 53, 54, 56 dan seterusnya
58	Mulai *bit* ke-1 sampai ke-6, 12, 20, 21, 35, 36, 50,51, 53, 54, 56 dan seterusnya
57	Mulai *bit* ke-1 sampai ke-8, 13, 20, 21, 35, 36, 37, 50-54, 56 dan seterusnya
56	Mulai *bit* ke-1 sampai ke-9, 14, 20, 21, 23, 35-38, 50-54, 56 dan seterusnya
55	Mulai *bit* ke-1 sampai ke-10, 15, 20, 21, 23, 24, 35-39, 50-54, 56 dan seterusnya
54	Mulai *bit* ke-1 sampai ke-11, 16, 20, 21, 23, 24, 25,

	35-40, 50 dan seterusnya
53	Mulai *bit* ke-1 sampai ke-12, 17, 20, 21, 23-26, 35-41, 50 dan seterusnya
52	Mulai *bit* ke-1 sampai ke-13, 18, 20, 21, 23-27, 35-42, 50 dan seterusnya
51	Mulai *bit* ke-1 sampai ke-14, 19, 20,21, 23-28, 35-43, 50 dan seterusnya
50	Mulai *bit* ke-1 sampai ke-15, 20, 21, 23-29, 35-44, 50 seterusnya
49	Mulai *bit* ke-1 sampai ke-16, 20, 21, 23-30, 35-45, 50 dan seterusnya
48	Mulai *bit* ke-1 sampai ke-17, 20, 21, 23-31, 35-46, 50 dan seterusnya
47	Mulai *bit* ke-1 sampai ke-18, 20, 21, 23-32, 35-47, 50 dan seterusnya
46	Mulai *bit* ke-1 sampai ke-21, 23-33, 35-48, 50 dan seterusnya
45	Mulai *bit* ke-1 sampai ke-21, 23 dan seterusnya
44	Selain mulai *bit* ke-22
43	Semua posisi

Tabel 4.5 di atas dibaca sebagai berikut:

Kolom pertama menunjukkan jumlah *bit error* mengelompok yang dapat dikoreksi oleh sistem dan kolom kedua menunjukkan syarat letak *bit error* mengelompok untuk jumlah yang bersesuaian pada kolom pertama. Misalnya pada baris hasil yang pertama dimana jumlah *bit error* sebanyak 65 *bit* dan syarat letaknya adalah mulai *bit* ke-5, 51, 53, 54, 56 dan seterusnya. Artinya kesalahan mengelompok sebanyak 65 *bit* dapat dikoreksi seluruhnya jika kesalahan tersebut letaknya dimulai dari *bit* ke-5 atau ke-51 atau ke-53 atau ke-54, atau ke-56 dan seterusnya. Begitu juga untuk baris kedua dan seterusnya. Sedangkan di baris terakhir yaitu untuk kesalahan mengelompok sebanyak 43 *bit* dapat dikoreksi seluruhnya pada semua posisi. Dengan kata lain, sistem ini dapat mengoreksi hingga 43 *bit* yang mengelompok di sembarang posisi, dan dapat diketahui pula bahwa letak kesalahan menjadi faktor yang cukup mempengaruhi hasil koreksi dari sistem gabungan teknik RS (15,5) dan BCH (15,5) ini.

4.2.2 Pola Kesalahan Menyebar

Pada pengujian dengan pola kesalahan menyebar ini, model yang digunakan hampir serupa dengan model yang digunakan pada pengujian sebelumnya. Bedanya terletak pada blok *channel.* Jika pada pengujian sebelumnya blok *channel* direkayasa untuk menciptakan kesalahan yang mengelompok, maka pada pengujian kali ini blok *channel* direkayasa untuk menciptakan kesalahan yang menyebar. Ada dua cara untuk menciptakan kesalahan yang menyebar. Pertama menggunakan blok saluran atau *channel* yang ada di *library* simulink untuk kesalahan menyebar secara acak, dalam kasus ini menggunakan blok saluran AWGN. Blok saluran ini menciptakan kesalahan yang menyebar dengan letak yang acak dan probabilitas kesalahan yang mendekati nilai tertentu. Artinya, letak dan jumlah kesalahan berubah-ubah secara acak menyerupai keadaan saluran yang sesungguhnya.

Cara kedua adalah menggunakan blok *channel* seperti pada pengujian dengan kesalahan mengelompok. Blok *channel* ini juga dapat dirancang untuk menciptakan kesalahan menyebar secara merata dengan jumlah dan letak yang tetap untuk tiap parameter yang ditentukan. *Channel* semacam ini memang jauh dari sifat saluran yang sebenarnya yang sukat diprediksi. Namun cara ini cukup berguna untuk mengetahui secara pasti kemampuan sistem *error correction* yang dirancang dalam mengatasi *error.*

Pengujian dengan cara pertama menggunakan saluran AWGN dengan tingkat kesalahan sekitar 0,186 dan dilakukan pemrosesan data acak sebanyak 20 *frame*. Hasilnya dapat dilihat pada Tabel 4.6 berikut.

Tabel 4.6 Hasil pengujian sistem dengan kesalahan menyebar menggunakan saluran AWGN

Frame ke-	Data Asli	Jumlah *Bit Error* / Persen *Error*	Data Hasil *Decoding*	Jumlah Simbol *Error* (akhir)
1	[7 0 3 13 5]	35 / 19,44	[7 0 3 13 5]	0
2	[4 9 15 3 9]	36 / 20,00	[6 0 11 3 9]	3
3	[5 10 2 15 3]	31 / 17,22	[5 10 2 15 3]	0
4	[5 4 4 5 11]	26 / 14,44	[5 4 4 5 11]	0
5	[0 1 15 13 7]	39 / 21,66	[0 1 14 15 7]	2
6	[5 15 12 6 1]	29 / 16,11	[5 15 12 6 1]	0

7	[12 6 15 2 11]	39 / 21,66	[4 14 13 4 11]	4
8	[8 9 13 1 14]	40 / 22,22	[8 9 13 1 14]	0
9	[15 11 9 12 5]	39 / 21,66	[2 3 11 6 5]	4
10	[13 2 7 13 13]	31 / 17,22	[13 2 7 13 13]	0
11	[4 5 3 0 15]	29 / 16,11	[4 5 3 0 15]	0
12	[15 6 11 9 10]	28 / 15,55	[15 6 11 9 10]	0
13	[7 0 15 8 5]	33 / 18,33	[7 0 15 8 5]	0
14	[8 1 15 4 4]	37 / 20,55	[8 1 15 4 4]	0
15	[10 4 11 8 5]	36 / 20,00	[10 4 11 8 5]	0
16	[0 7 9 13 11]	35 / 19,44	[0 7 9 13 11]	0
17	[6 8 6 0 10]	37 / 20,55	[14 11 2 0 0]	4
19	[5 4 11 4 7]	30 / 16,66	[5 4 11 4 7]	0
20	[13 5 3 5 2]	40 / 22,22	[13 5 3 4 2]	1
26	[4 4 0 1 2]	35 / 19,44	[14 4 2 9 2]	3

Pada Tabel 4.6 di atas dapat dilihat ada beberapa data *frame* dengan jumlah *bit error* yang sama namun hasil *decoding*-nya berbeda. Misalnya pada *frame* ke-2 dan ke-15, *frame* ke-8 dan ke-20, serta *frame* ke-16 dan ke-26. Jika ditinjau lebih jauh, perbedaan hasil *decoding* pada ketiga pasang *frame* tersebut disebabkan oleh perbedaan letak *bit error* yang dihasilkan oleh *channel.* Perbedaan letak *bit error* tersebut dapat dilihat pada Tabel 4.7 berikut.

Tabel 4.7 Tinjauan letak *bit error* menyebar

Frame ke-	Jumlah *bit error*	Jumlah simbol *error*	Letak urutan *bit error*
2	36	3	[3 5 7 8] [18 20 23 30] (34 41 42) (47 48 49) [64 66 70 73] (87) [91 97 103 104] (116) [123 127 130 133] (136 138) [157 160 161 162] (168 173)
15	36	0	[4 5 7 12 13] [18 19 23 25 27] (38 40 41) (50 55) ( ) (79 83 87) (102 104) (108 113 118) (126 132) (136) [151 154 156 158 161 163 165] (168 171 172)
8	40	0	(2 11) [19 22 24 28] (34 36 41) (49 55) (69 73) [80 82 84 87 89] [91 92 99 102 105] (108 110 115) [122 126 127 134 135] [136 146 147 148] (156 160 161) (170 179)
20	40	1	(6 9 15) (20 26 29) (43) [46 51 53 54 58] [61 63 69 72 73] [78 80 83 88 89] [95 97 99 105] (106 116 119) ( ) (141

			147 150) [152 156 157 161] [168 169 170 174]
16	35	0	(2 11) (16 21) (36 42) (47 52 58) (66 73) (89) (97 99 100) [109 114 115 116 118] [121 123 125 127 129 130] (149 150) (151 152 158) [167 170 171 175]
26	35	3	[1 3 8 11 14] (26 29 30) [31 33 37 45] (51 54) [64 65 68 69 70] [81 84 85 89] [91 99 100 101 103] (109 113 120) (135) (141) (151 158)

Kolom ”Letak urutan *bit error*” pada Tabel 4.7 berisi urutan *bit* yang mengalami *error* dan terbagi ke dalam 12 segmen berisi *bit error* dari tiap 15 *bit* kata sandi yang akan di-*decode*. Isi dalam kurung pertama menujukkan *bit error* pada 15 *bit* pertama, kurung kedua menunjukkan *bit error* pada *bit* ke16 sampai 30 dan seterusnya sampai kurung ke-12 pada 15 *bit* ke-12. Pada proses *decoding* BCH (15,5), jumlah *error* yang dapat dikoreksi adalah 3 *bit* dari tiap 15 *bit* yang di-*decode*. Kurung siku menunjukkan jumlah *error* lebih dari 3 *bit* dalam 15 *bit*, sehingga akan menyisakan *error* pada hasil *decoding* oleh *decoder* BCH (15,5).

Pada *frame* kedua terdapat 6 segmen dengan *error* lebih dari 3 *bit,* sehingga akan menyisakan *error* di 6 segmen dari hasil *decoding* BCH (15,5). *Error* di 6 segmen ini menghasilkan *error* 8 simbol setelah dikonversi dari *bit* ke *integer*. Sisa *error* 8 simbol menyebar ini tentunya tidak akan dapat dikoreksi dengan baik oleh *decoder* RS (15,5), dan pada hasil akhirnya masih menyisakan 3 simbol *error*. Sedangkan pada *frame* ke-15 hanya terdapat 3 segmen dengan *error* lebih dari 3 *bit*, sehingga menyisakan *error* di 3 segmen dari hasil *decoding* BCH (15,5). Kemudian setelah dikonversi ke *integer*, menghasilkan 5 simbol *error*. Sisa *error* 5 simbol menyebar ini dapat dikoreksi dengan baik oleh *decoder* RS (15,5), dan pada hasil akhirnya sudah tidak menyisakan *error*.

Dalam pengujian dengan kesalahan menyebar ini, penyandian BCH (15,5) memegang peranan penting karena jika pada tahap *decoding* oleh *decoder* BCH (15,5) gagal mengoreksi kesalahan dan masih menyisakan kesalahan yang menyebar melebihi kemampuan dasar *decoder* RS (15,5), kemungkinan besar tahap *decoding* selanjutnya oleh *decoder* RS (15,5) juga gagal mengoreksi kesalahan yang ada. Namun jika hasil proses *decoding* oleh *decoder* BCH (15,5) masih menyisakan kesalahan yang mengelompok ataupun menyebar dalam batas kemampuan dasar *decoder* RS (15,5), maka tahap *decoding* selanjutnya oleh

decoder RS (15,5) masih besar kemungkinan untuk dikoreksi dengan baik. Dari hasil pengujian, dapat diambil rata-rata untuk jumlah *error* menyebar yang dapat dikoreksi dengan baik oleh sistem gabungan teknik BCH (15,5) dan RS (15,5) ini adalah sekitar 36 *bit* atau 20 persen sesuai dengan kemampuan penyandian BCH (15,5). Dengan asumsi bahwa kesalahan menyebar hanya mempunyai satu pola yaitu merata dan tidak ada yang mengelompok.

4.3 PENGUJIAN DENGAN DSK TMS320C6713

Pengujian dengan DSK TMS320C6713 ini bertujuan untuk melakukan simulasi terhadap sistem yang telah diterapkan pada sebuah perangkat keras, yaitu papan DSK TMS320C6713. Untuk melihat hasil kerja perangkat DSK menggunakan alat peraga *oscilloscop* agak sulit dilakukan sebab sinyal yang diamati bukanlah sinyal kontinu dan hanya muncul sesaat. Oleh karena itu digunakanlah blok "*To RTDX*" sebagai penghubung antara DSK dan program Matlab dalam pertukaran data dari DSK ke Matlab. Sumber pesan yang digunakan bukan berupa data acak dari pembangkit sinyal seperti *random interger generator*, melainkan data diambil dari *file* berformat {.m} dengan nama "data.m" dan nama variabel datanya "input_data". *File* data ini berisi 15 simbol heksadesimal yang kemudian dapat dimuat ke *workspace* Matlab dan dapat diakses sebagai oleh model simulink menggunakan blok sumber "*from workspace"* dalam bentuk *frame* berisi 5 simbol, sehingga dari *file* data tersebut bisa diambil 3 *frame* data 5 simbol.

Langkah pengujiannya sebagai berikut:

1. Data sumber dimasukkan ke *workspace* Matlab dengan cara mengeksekusi *file* data.m pada *Command Window* Matlab, maka akan muncul variabel "input_data" pada *workspace*.
2. Model rangkaian untuk uji dengan RTDX seperti pada Gambar 3.12 diaktifkan. Setelah perangkat DSK terhubung ke PC/laptop dengan baik dan program *Code Composer Studio* juga sudah diaktifkan, maka dilakukan tahap *Incremental Build*, yaitu membangun program bahasa C dari model simulink yang dibuat dan memasukkannya ke dalam memori DSK sebagai kode program yang siap dijalankan.

3. Program *driver* untuk RTDX dijalankan melalui *Command Window* Matlab. Program tersebut berupa perintah-perintah yang tersimpan dalam *file* berformat .m berungsi untuk menjalankan program yang sudah terpasang di DSK, membuka jalur RTDX, melakukan pertukaran data dari DSK ke Matlab dan menghentikan jalannya program. Dalam program *driver* juga ditentukan lama proses berjalannya program di DSK dan pertukaran datanya dengan Matlab. Bentuk program *driver* RTDX dapat dilihat pada lampiran.
4. Setelah program dalam DSK selesai bekerja, data yang telah dimuat dari DSK melalui RTDX disimpan pada *directory* yang telah ditentukan dengan format {.mat} dan nama variabel yang juga sudah ditentukan pada program *driver*.

Untuk pengujian DSK dengan RTDX ini ada 3 macam model serupa namun berbeda pola kesalahan, yaitu pola mengelompok, pola menyebar yang tetap, dan pola menyebar acak mengunakan saluran BSC (*Binary Symmetric Channel*). Tiap model ditempatkan pada *folder* yang berbeda dan masing-masing memiliki 4 *file* program *driver* sesuai dengan jumlah blok "*to* RTDX" dalam sebuah model yang berjumlah 4. Keempat program *driver* tersebut masing-masing untuk mengambil data antara lain pesan awal, pesan dikirim, pesan diterima, dan pesan akhir. Data yang diambil melalui RTDX dapat dilihat pada bagian lampiran.

4.4 PERBANDINGAN HASIL UJI DSK DENGAN RTDX TERHADAP HASIL UJI SIMULINK

Untuk membandingkan hasil uji menggunakan DSK dengan RTDX terhadap hasil uji menggunakan simulink, maka dilakukan pengujian sekali lagi menggunakan simulink dengan parameter yang sama seperti pada uji DSK. Parameter yang disamakan antara lain sumber pesan dan saluran. Sumber yang digunakan yaitu menggunakan blok *"signal from workspace"*. Data yang akan diproses berupa rangkaian 15 bilangan *integer* yang terbagi ke dalam 3 *frame*. Sedangkan salurannya menggunakan blok BSC (*Binary Symmetric Channel*). Hasil pengujian menggunakan Simulink disajikan pada Tabel 4.8.

Tabel 4.8 Data hasil uji Simulink dengan parameter yang sama dengan uji DSK

***Frame* ke-**	**1**
Pesan awal	15 7 10 2 9
Pesan dikirim	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1
Pesan diterima	1 1 1 1 ***1 0 1 0*** 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 ***0*** 1 0 1 0 0 0 1 1 1 1 ***0*** 0 1 1 0 ***1*** 0 1 ***1*** 0 1 1 0 1 1 1 0 0 ***1*** 0 1 ***1*** 1 ***0*** 0 1 1 1 ***1 1*** 0 0 1 0 1 ***1*** 1 ***1 1*** 1 1 0 1 1 1 0 0 0 0 ***0 1*** 1 1 1 0 ***1*** 0 0 1 0 ***0*** 0 ***1 0*** 1 0 0 1 0 ***0 1*** 0 1 1 0 1 1 ***0*** 0 0 0 0 1 ***0*** 1 0 0 ***1*** 0 ***0 1*** 1 0 0 1 1 ***1*** 0 0 1 1 0 1 1 ***0*** 0 0 0 0 1 0 ***0 1*** 0 1 ***0*** 0 1 ***0*** 1 0 0 0 0 1 0 1 0 0 1 1 0 ***0*** 1 1 0 0 0 0 1 0 ***0***
Pesan akhir	15 ***15*** 10 2 9
***Frame* ke-**	**2**
Pesan awal	13 12 4 6 1
Pesan dikirim	1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0
Pesan diterima	1 1 0 1 1 1 0 0 0 0 ***0*** 0 1 ***1*** 0 1 0 0 ***1*** 1 1 1 1 0 ***0*** 0 1 1 0 0 0 0 0 1 1 1 1 0 1 ***1*** 1 1 0 0 1 0 0 0 ***1*** 1 0 ***0*** 0 0 1 ***0*** 0 1 ***0 0*** 1 ***0*** 0 0 1 0 0 0 1 1 1 1 ***1*** 1 0 0 1 0 1 ***0*** 0 0 1 0 0 0 1 1 1 1 0 0 ***1*** 0 0 0 0 0 ***1*** 0 0 0 ***1*** 0 0 0 0 1 1 0 ***0*** 1 1 0 0 0 0 0 ***0*** 0 1 ***0*** 1 0 ***0*** 0 1 1 ***1 1*** 0 0 0 ***1*** 1 1 0 ***0*** 1 0 ***1*** 1 ***1*** 1 0 1 1 ***0*** 1 0 ***0*** 1 1 1 ***1*** 0 0 0 1 0 1 0 0 1 1 ***1*** 0 1 ***0*** 1 1 0 1 0 1 1 0 ***1 0*** 0 0
Pesan akhir	13 12 4 6 1
***Frame* ke-**	**3**
Pesan awal	14 5 8 11 3
Pesan dikirim	1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
Pesan diterima	1 ***0*** 1 0 0 ***1*** 0 1 0 1 0 0 1 1 0 ***0*** 0 1 1 ***1*** 0 1 ***1*** 0 0 1 1 1 1 0 0 0 1 ***1*** 1 0 0 1 ***0*** 0 ***0*** 1 1 0 0 1 0 0 1 1 0 1 1 1 ***1*** 0 0 0 1 ***1*** 0 ***1 1*** 1 0 1 ***1*** 0 1 1 0 1 1 1 0 0 ***1*** 1 1 0 ***0 0*** 1 0 0 0 0 1 ***1 0 1 1*** 0 ***0*** 0 1 0 0 1 ***0*** 0 1 1 1 ***1 0*** 0 ***1*** 1 0 0 0 1 1 ***0*** 1 ***1*** 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 ***1 1*** 0 0 1 0 ***0*** 0 0 1 1 0 1 ***0 0*** 1 1 1 1 ***0*** 1 ***0 0*** 1 1 1 1 1 1 1 1 1 0 ***0*** 0 ***0*** 1 0 ***1*** 1 1 ***1 1*** 0 1
Pesan akhir	14 5 8 11 3

Terlihat pada Tabel 4.8 di atas, data yang tercetak miring dan berwarna merah merupakan data yang mengalami kesalahan. Setelah membandingkan data hasil pengujian menggunakan DSK yang diambil melalui RTDX terhadap hasil pengujian menggunakan Simulink, terlihat bahwa program yang diterapkan pada DSK telah bekerja dan hasil kerja DSK tersebut sesuai dengan hasil simulasi yang dilakukan pada simulink.

BAB 5
KESIMPULAN

Berdasarkan hasil pengujian dan analisis sistem pada Bab 4, dapat diambil beberapa kesimpulan sebagai berikut:

1. Metode penyandian siklis BCH dan RS dapat dikombinasikan secara serial dan mampu memperbaiki kekurangan dari masing-masing teknik dalam mengoreksi kesalahan.
2. Secara khusus, penggabungan teknik penyandian RS (15,5) dengan BCH (15,5) mampu mengoreksi maksimum 116 *bit* kesalahan mengelompok di belakang dan 43 *bit* kesalahan mengelompok di posisi manapun serta sekitar 36 *bit* kesalahan menyebar.
3. Letak kesalahan memengaruhi hasil koreksi dari sistem dengan penggabungan teknik RS (15,5) dan BCH (15,5), dikarenakan *decoder* BCH (15,5) menerima masukan data *frame* dengan panjang 60 *bit* dan adanya proses konversi dari *bit* ke *integer* pada tahapan *decoding*.
4. Model simulink rangkaian *error correction* dengan penggabungan teknik RS (15,5) dan BCH (15,5) secara serial telah berhasil diterapkan dan disimulasikan pada papan DSK TMS320C6713 dengan fungsi dan hasil yang sama dengan simulasi menggunakan simulink.

DAFTAR REFERENSI

[1] Lin, Shu., Daniel J. Costello, Jr., *Error Control Coding: Fundamentals and Applications,* Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1983.

[2] Peterson, W. Wesley, E. J. Weldon. Jr, *Error-Correcting Codes*, Colonial Press Inc, United States of America, 1972.

[3] Moreira, J.C. dan Farrell, P.G., *Essentials of Error-Control Coding,* John Wiley & Sons Ltd, Chichester, 2006.

[4] Reed, Irving S., Chen, Xuemin, *Error-Control Codes for Data Network.*

[5] Sylvester, Joey, *Reed Solomon Codes*, Elektrobit, 2001.

[6] Murdocca, M.J. dan Heuring, V.P., *Principles Of Computer Architecture,* Prentice Hall, 1999.

[7] "____", *Introduction to Simulink, Link for CCS & Real-Time Workshop*, 2006. Diakses dari: http://www.emba.uvm.edu/~mirchand/classes/EE275/2007/Real-Time/Lab6.pdf, 8 Juni 2010

LAMPIRAN A

Program *Driver* 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', 100); % Reset timeout = 10 seconds
rx.configure(64000,2);
rx.open('pesan_awal','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=21;
i=1;
enable(rx,'pesan_awal');
while (i<ukuran)
      if isenabled(rx,'pesan_awal')
          pesanawal(i,1)=readmsg(cc.rtdx,'pesan_awal','double');
      end
    i=i+1;
end

RTDXcleanup(cc,rx);
            matfile=strcat('pesanmula','.mat');
            save(matfile,'pesanawal');
%=========================================================================
% 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,'pesan_awal');
disable(rx);                % disable RTDX
close(cc.rtdx,'pesan_awal');
```

LAMPIRAN B

Tabel Representasi α dari elemen GF(2^3) dibangkitkan oleh $p(x) = 1 + x + x^3$

Power Representasi	Polinomial Representasi	3-bit Representasi		
		b0	b1	b2
0	0	0	0	0
1	1	1	0	0
α	α	0	1	0
α^2	α^2	0	0	1
α^3	$\alpha + 1$	1	1	0
α^4	$\alpha^2 + \alpha$	0	1	1
α^5	$\alpha^2 + \alpha + 1$	1	1	1
α^6	$\alpha^2 + 1$	1	0	1

LAMPIRAN C
Data Dari RTDX

Kesalahan 60 *bit* mengelompok di depan

***Frame* ke-**	**1**
Pesan awal	15 7 10 2 9
Pesan dikirim	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1
Pesan diterima	***0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 1 1 1 1 0*** 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1
Pesan akhir	15 7 10 2 9

***Frame* ke-**	**2**
Pesan awal	13 12 4 6 1
Pesan dikirim	1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0
Pesan diterima	***0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1*** 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0
Pesan akhir	13 12 4 6 1

***Frame* ke-**	**3**
Pesan awal	14 5 8 11 3
Pesan dikirim	1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
Pesan diterima	***0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0*** 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
Pesan akhir	14 5 8 11 3

Kesalahan 60 *bit* menyebar secara merata

***Frame* ke-**	**1**
Pesan awal	15 7 10 2 9
Pesan dikirim	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1
Pesan diterima	***0 0 0 0*** 0 1 0 1 1 0 0 1 0 0 0 ***0 0 0 0*** 0 1 0 1 1 0 0 1 0 0 0 ***0 1 1 1*** 1 1 1 1 0 1 0 1 1 0 0 ***1 0 1 1*** 1 1 0 1 1 1 0 0 0 0 1 ***1 0 0*** 0 1 1 1 0 0 0 0 1 0 1 0 ***0 1 1*** 1 1 0 1 1 1 0 0 0 0 1 0 ***0 0 0*** 0 0 0 0 1 0 1 0 0 1 1 0 ***1 0 1*** 1 0 0 1 1 0 1 1 1 0 0 0 ***0 0 0*** 0 0 0 0 1 0 1 0 0 1 1 0 ***1 1 0*** 1 0 1 1 1 0 0 0 0 1 0 1 ***1 1 0*** 1 0 1 1 1 0 0 0 0 1 0 1 ***1 1 0*** 1 0 1 1 1 0 0 0 0 1 0 1
Pesan akhir	15 7 10 2 9

***Frame* ke-**	**2**
Pesan awal	13 12 4 6 1
Pesan dikirim	1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0
Pesan diterima	***0 0 1 0*** 1 1 0 0 0 0 1 0 1 0 0 ***0 1 1 1*** 1 1 1 1 0 1 0 1 1 0 0 ***1 1 1 0*** 1 1 1 0 1 0 1 1 0 0 1 ***1 1 1 1*** 1 0 1 0 0 1 1 0 1 1 1 ***0 0 1*** 0 1 0 0 0 1 1 1 1 0 1 0 ***1 0 1*** 1 1 0 0 1 0 0 0 1 1 1 1 ***1 1 1*** 0 0 0 0 0 0 0 0 0 0 0 0 0 ***1 1 0*** 1 0 1 1 1 0 0 0 0 1 0 1 ***0 0 1*** 1 0 1 1 0 0 1 0 0 0 1 1 ***1 0 0*** 0 0 1 0 0 0 1 1 1 1 0 1 ***0 0 0*** 0 0 0 0 1 0 1 0 0 1 1 0 ***1 0 0*** 1 1 0 1 0 1 1 0 0 1 0 0
Pesan akhir	13 12 4 6 1

***Frame* ke-**	**3**
Pesan awal	14 5 8 11 3
Pesan dikirim	1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
Pesan diterima	***0 0 0 1*** 0 0 0 1 0 1 0 0 1 1 0 ***0 1 0 0*** 0 0 1 0 0 0 1 1 1 1 0 ***1 1 0 1*** 1 0 0 1 1 0 1 1 1 0 0 ***0 1 1 0*** 1 0 1 1 1 0 0 0 0 1 0 ***1 1 1*** 1 0 1 0 0 1 1 0 1 1 1 0 ***1 1 0*** 1 0 1 1 1 0 0 0 0 1 0 1 ***1 1 1*** 1 0 1 0 0 1 1 0 1 1 1 0 ***0 1 1*** 1 0 0 0 1 1 1 1 0 1 0 1 ***1 0 1*** 0 0 1 1 1 1 0 1 0 1 1 0 ***0 1 1*** 0 0 1 0 1 0 0 1 1 0 1 1 ***0 0 0*** 1 1 1 1 1 1 1 1 1 1 1 1 ***0 0 0*** 0 1 0 1 1 0 0 1 0 0 0 1
Pesan akhir	14 5 8 11 3

Kesalahan menyebar secara acak

***Frame* ke-**	**1**
Pesan awal	15 7 10 2 9
Pesan dikirim	1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1
Pesan diterima	1 1 1 1 ***1 0 1 0*** 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 ***1*** 0 1 0 1 0 0 0 1 1 1 1 ***0*** 0 1 1 0 ***1*** 0 1 ***1*** 0 1 1 0 1 1 1 0 0 ***1*** 0 1 ***1*** 1 ***0*** 0 1 1 1 ***1 1*** 0 0 1 0 1 ***1*** 1 ***1 1*** 1 0 1 1 1 0 0 0 0 ***0 1*** 1 1 1 0 ***1*** 0 0 1 0 ***0*** 0 ***1 0*** 1 0 0 1 0 ***0 1*** 0 1 1 0 1 1 ***0*** 0 0 0 0 1 ***0*** 1 0 0 ***1*** 0 ***0 1*** 1 0 0 1 1 ***1*** 0 0 1 1 0 1 1 ***0*** 0 0 0 0 0 1 0 ***0 1*** 0 1 ***0*** 0 1 ***0*** 1 0 0 0 0 1 0 1 0 0 1 1 0 ***0*** 1 1 0 0 0 0 1 ***0***
Pesan akhir	15 ***15*** 10 2 9

***Frame* ke-**	**2**
Pesan awal	13 12 4 6 1
Pesan dikirim	1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0
Pesan diterima	1 1 0 1 1 1 0 0 0 0 ***0*** 0 1 ***1*** 0 1 0 0 ***1*** 1 1 1 1 ***0*** 0 1 1 0 0 0 0 0 1 1 1 1 0 1 ***1*** 1 1 0 0 1 0 0 0 ***1*** 1 ***0*** 0 0 1 ***0*** 1 ***0*** 0 ***1*** 0 0 0 1 0 0 0 1 1 1 1 ***1*** 1 0 0 1 0 1 ***0*** 0 0 1 0 0 0 1 1 1 1 0 0 ***1*** 0 0 0 0 0 ***1*** 0 0 0 ***1*** 0 0 0 0 1 1 ***0*** 0 1 1 0 0 0 0 ***0*** 0 1 ***0*** 1 0 ***0*** 0 1 1 ***1 1 0*** 0 0 ***1*** 1 1 0 ***0*** 1 0 ***1*** 1 ***1 1*** 0 1 1 ***0*** 1 0 ***0*** 1 1 1 ***1*** 0 0 0 1 0 1 0 0 1 1 ***1*** 0 1 ***0*** 1 1 0 1 0 ***1*** 1 0 ***1 0*** 0 0
Pesan akhir	13 12 4 6 1

***Frame* ke-**	**3**
Pesan awal	14 5 8 11 3
Pesan dikirim	1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1
Pesan diterima	1 ***0*** 1 0 0 ***1*** 0 1 0 1 0 0 1 1 0 ***0*** 0 1 1 ***1*** 0 1 ***1*** 0 0 1 1 1 1 0 0 0 1 ***1*** 1 0 0 1 ***0*** 0 ***0*** 1 1 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 ***1*** 0 0 0 1 ***1*** 0 1 ***1*** 1 0 1 1 0 1 1 1 0 0 ***1*** 1 1 0 ***0 0*** 1 0 0 0 0 1 ***0*** 1 ***1 1*** 0 1 ***1 1*** 0 ***0*** 0 1 0 0 1 ***0*** 0 1 1 1 ***1 0*** 0 ***1*** 1 0 0 0 1 1 ***0*** 1 ***1*** 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 ***1 1*** 0 0 1 0 ***0*** 0 0 1 1 0 1 ***0 0*** 1 1 1 1 ***0*** 1 ***0 0*** 1 1 1 1 1 1 1 1 1 0 ***0 0*** 0 1 0 ***1*** 1 ***1 1*** 0 1
Pesan akhir	14 5 8 11 3