RANCANG BANGUN CODEC REED-SOLOMON BERBASIS FPGA PADA KOMUNIKASI DATA LEWAT KANAL RADIO

PRIHADI MURDIYAT
NRP. 2201 203 013

PROGRAM STUDI MAGISTER
BIDANG KEAHLIAN TELEKOMUNIKASI MULTIMEDIA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

RANCANG BANGUN CODEC REED-SOLOMON BERBASIS FPGA PADA KOMUNIKASI DATA LEWAT KANAL RADIO

Tesis ini disusun untuk memenuhi salah satu syarat
memperoleh gelar Magister Teknik (M.T.)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
Prihadi Murdiyat
NRP. 2201 203 013

Tanggal Ujian : 28 Januari 2005
Periode Wisuda : 2005

Disetujui oleh Tim Penguji Tesis :

1. **Dr. Ir. Titon Dutono,M Eng** — **(Pembimbing I)**
   NIP. 131 651 258
2. **Ir. Suwadi, MT.** — **(Pembimbing II)**
   NIP. 132 053 513
3. **Ir. Achmad Ansori, DEA** — **(Penguji)**
   NIP. 131 855 878
4. **Ir. Endroyono, DEA.** — **(Penguji)**
   NIP. 131 943 644
5. **Dr. Ir. Gamantyo H.,M Eng** — **(Penguji)**
   NIP. 132 048 147
6. **Dr. Ir. Wirawan, DEA.** — **(Penguji)**
   NIP. 131 842 501

Direktur Program Pascasarjana

Prof. Ir. Happy Ratna S., M.Sc PhD
NIP. 130 541 829

RANCANG BANGUN CODEC REED-SOLOMON BERBASIS FPGA PADA KOMUNIKASI DATA LEWAT KANAL RADIO

Nama mahasiswa : **Prihadi Murdiyat**
NRP. : **2201 203 013**
Pembimbing : **Dr. Ir. Titon Dutono,M Eng.**
Co-Pembimbing : **Ir. Suwadi, MT.**

ABSTRAK

Pada transmisi data digital, noise pada kanal dapat menyebabkan kesalahan penerimaan data. Untuk mengatasinya, dapat digunakan pengkodean kanal. Pada tesis ini, jenis kode yang digunakan adalah kode Reed Solomon RS(15,11,2) yang merupakan jenis kode FEC (Forward Error Correction) dengan elemen simbol yang diambil dari Galois Field GF(2^4). Penulisan RS(15,11,2) menunjukkan bahwa kode ini menerima informasi dengan panjang 11 simbol (dengan lebar simbol = 4 bit) dan mengubahnya menjadi codeword dengan panjang 15 simbol. Secara teoritis, kode ini mampu memperbaiki kesalahan simbol maksimal sebesar t = 2.

Setelah skema encoder dan decoder RS dirancang, implementasi dilakukan pada FPGA (Field Programmable Gate Array) Xilink XC4010XLP84. Hasil pengujian menunjukkan bahwa encoder mengeluarkan codeword sesuai codeword perhitungan. Sementara itu, decoder mampu memperbaiki kesalahan simbol maksimal 2 buah. Decoder juga mampu memperbaiki kesalahan akibat burst, selama kerusakan yang ditimbulkan burst tidak melebihi 2 simbol.

Hasil evaluasi pada kanal radio FM 100 Mhz, menunjukkan bahwa codec dapat mengurangi kesalahan akibat penurunan ratio sinyal terima (dBμV) yang identik dengan menurunnya SNR (Signal to Noise Ratio). Atau dengan kata lain adanya codec dapat memperbaiki $P_{e\ simbol}$ (Probability of symbol error).

Kata kunci : Reed-Solomon, Galois Field, Field Programmable Gate Array, SNR, $P_{e\ simbol}$

DESIGNING AND IMPLEMENTING REED-SOLOMON CODEC BASED ON FPGA FOR DATA COMMUNICATION IN RADIO CHANNEL

Student name	**: Prihadi Murdiyat**
NRP.	**: 2201 203 013**
Supervisor	**: Dr. Ir. Titon Dutono,M Eng.**
Co-Supervisor	**: Ir. Suwadi, MT.**

ABSTRACT

In digital data transmission, channel noise causes error in data reception. In order to decrease the error, channel coding could be used. This thesis uses Reed-Solomon RS (15,11,2) code which is type of FEC (Forward Error Correction) whose element symbols are taken from Galois Field GF(2^4). The notation RS(15,11,2) shows that this code converse 11 symbols of information (at 4 bit period) and changes it into 15 symbol codeword. Theoretically, this code is able to correct t = 2 symbols maximum.

When encoder and decoder circuits were designed, the implementation is performed to FPGA (Field Programmable Gate Array) Xilink XC4010XLP84. The test results that the encoder gives the codeword which same with the calculation codeword. In other hand, the decoder has ability to correct up to 2 symbols. The decoder could be correct burst error whose damage up to 2 symbols.

The evaluation in FM 100 MHz radio channel shows that the codec could be decrease errors whose rises when received signal ratio (dBμV) is decrease that identically with SNR (Signal to Noise Ratio) decrease. In other word, the codec could be correct the $P_{e\ symbol}$ (Probability of symbol error).

Keyword : Reed-Solomon, Galois Field, Field Programmable Gate Array, SNR, $P_{e\ symbol}$

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT, karena hanya berkat rahmat dan hidayahNya saya dapat menyelesaikan tesis dengan judul Rancang bangun Reed-Solomon berbasis FPGA pada komunikasi data lewat kanal radio, sebagai salah satu persyaratan untuk memperoleh gelar Magister Teknik di Program Pasca Sarjana ITS Surabaya.

Atas selesainya tesis ini, saya juga menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Bapak Direktur dan Staf Politeknik Negeri Samarinda, atas pemberian kesempatan dan bantuan dalam melanjutkan studi di ITS Surabaya.
2. Staf pengajar jurusan Teknik Elektro FTI-ITS, yang telah memberi ilmu bermanfaat
3. Bapak Dr. Ir. Titon Dutono selaku dosen pembimbing I, atas saran dan dukungannya yang menjadi dorongan untuk menyelesaikan tugas ini
4. Bapak Ir. Suwadi, MT selaku dosen pembimbing II dan dosen wali, atas bimbingan dan bantuannya selama kuliah maupun saat mengerjakan tesis.
5. Sdr. Hendro Hermanto yang telah menjadi kawan seperjuangan dalam mengerjakan tugas-tugas maupun tesis ini.
6. Ka Prodi dan rekan-rekan widya iswara prodi Elektronika VEDC Malang yang memberikan banyak masukan, serta kemudahan pemakaian fasilitas untuk penelitian tesis ini.
7. Istri dan anak-anak tercinta yang memberikan dorongan moril, serta merelakan ayahnya untuk beberapa waktu sibuk menyelesaikan tugas berat ini.
8. Serta rekan-rekan lain yang tidak mungkin disebutkan semua.

Semoga Allah SWT membalas semua bantuan yang telah diberikan, baik selama studi maupun dalam penyusunan tesis.

Sebaik apapun penelitian tesis ini, harus diakui bahwa di dalamnya masih terdapat kekurangan atau kesalahan yang harus diperbaiki. Untuk itu mohon kritik dan saran yang membangun, demi lebih baiknya penelitian-penelitian berikutnya. Semoga persembahan yang kecil ini dapat berguna bagi kita semua. Amin.

Teknik Elektro FTI ITS
Surabaya, Januari 2005.

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang

Saat ini, di Kalimantan Timur banyak terdapat industri dan instansi pemerintah yang mempunyai lokasi-lokasi (kantor, kapal, pabrik, areal penguasaan hutan dan area pertambangan) terpisah satu sama lain, dengan jarak beberapa kilometer hingga ratusan kilometer. Komunikasi antar lokasi dilakukan dengan berbagai cara seperti :

- telepon saluran tetap (*fixed line*)
- telepon seluler
- komunikasi satelit
- radio komunikasi pada *band* HF, VHF

Untuk komunikasi data, biasanya digunakan satelit VSAT (dilakukan oleh PMA bermodal besar), atau *fixed line* yang hanya dapat dilakukan jika lokasi perusahaan terletak di tempat yang terjangkau oleh layanan PT. Telkom. Karena itu, pengenalan dan pengembangan komunikasi data lewat kanal radio merupakan hal yang penting, mengingat sistem komunikasi ini dapat diterapkan dengan memanfaatkan perangkat komunikasi radio HF atau VHF yang ada.

Pengembangan komunikasi data lewat kanal radio perlu didukung dengan usaha-usaha peningkatan unjuk kerja sistem. Hal itu perlu dilakukan mengingat kanal radio mengalami banyak gangguan seperti : interferensi dari frekuensi lain, distorsi akibat *multipath*, pengaruh medan magnet atau medan listrik yang lebih

besar, dan lain-lain. Salah satu usaha yang ingin dilakukan penulis adalah merancang bangun *codec* (*coder-decoder*) Reed-Solomon berbasis FPGA (*Field Programmable Gate Array*) untuk diterapkan pada sistem komunikasi data lewat kanal radio.

Kode Reed-Solomon (RS) merupakan jenis kode blok yang cukup banyak digunakan. Kode ini bekerja berdasarkan Galois Field (GF) dan mempunyai kemampuan *Forward Error Correction* (FEC). Kemampuan FEC sangat diperlukan pada komunikasi digital, karena dapat menurunkan *probability of error*, mengurangi proses *Automatic Repeat Request* (ARQ) dan menambah jarak transmisi. Kode RS banyak digunakan pada aplikasi-aplikasi :

- media penyimpan (pita, CD, DVD, *barcodes*)
- komunikasi nirkabel (*wireless*) atau bergerak (*mobile*)
- komunikasi satelit
- *digital video broadcasting* (DVB)
- modem kecepatan tinggi
- dan aplikasi lain

Penggunaan kode RS yang semakin meluas menyebabkan timbulnya implementasi kode RS dalam berbagai bentuk. Penerapannya bisa dalam bentuk perangkat keras atau perangkat lunak.

Dalam bentuk perangkat keras, kode RS dapat disusun dari : rangkaian elektronik digital : diskrit, terpadu, atau rangkaian terpadu yang terprogram (*programmable*). Dengan rangkaian terpadu dan terprogram *Field Programmable Gate Array* (FPGA), rancang bangun *codec* RS menjadi lebih fleksibel dibanding menggunakan rangkaian-rangkaian elektronik diskrit dan non *programmable*.

1.2. Perumusan Masalah

Pada tesis ini, akan ditunjukkan proses rancang bangun dan pengujian *codec* hasil rancang bangun. Kemudian, hasil rancang bangun dievaluasi pada komunikasi data lewat kanal radio. Yang akan menjadi masalah dalam proses ini adalah :

a) **Pemilihan parameter m, t,** . Jumlah simbol m dipilih berdasar kemampuan port paralel LPT 1: komputer yang berfungsi sebagai sumber dan penerima data. Sedangkan t dipilih dengan pertimbangan mengurangi kerumitan rangkaian, agar proses *encoding* dan *decoding* dapat berlangsung lebih cepat.

b) **Penghitungan dan pengaturan waktu proses di tiap bagian**.

c) **Sinkronisasi *clock encoder* dengan sumber data.**

d) **Sinkronisasi *encoder* di pemancar dengan *decoder* di penerima**

e) **Apakah *codec* mempunyai kemampuan FEC sesuai teori ?**

f) **Apakah pemasangan *codec* hasil rancang bangun pada komunikasi lewat kanal radio menyebabkan perbaikan $P_{e\ simbol}$?**

Agar pembahasan tidak melebar, maka ditentukan batasan-batasan masalah sebagai berikut :

a) Frekuensi radio yang digunakan adalah 100 MHz dengan menggunakan FM stereo

b) Modem radio menggunakan IC TCM 3105 dengan modulasi AFSK.

c) Hanya menghasilkan satu macam *codec* RS dengan parameter tertentu

d) Untuk evaluasi, sistem komunikasi data yang digunakan adalah *point to point* satu arah.

1.3. Tujuan Penelitian

Tujuan pertama dari penelitian ini adalah mengimplementasikan *codec* RS pada FPGA dengan spesifikasi berdasar parameter yang dirancang. Tujuan berikutnya adalah membuktikan bahwa dengan dipasangnya *codec* RS pada komunikasi data lewat kanal radio, maka unjuk kerjanya meningkat.

Hasil penelitian diharapkan dapat digunakan untuk pengenalan dan pengembangan komunikasi data khususnya di Kalimantan Timur dengan aplikasi:

- komunikasi data untuk keperluan administrasi antar lokasi perusahaan
- komunikasi data untuk keperluan pengaturan jarak jauh terhadap *plant* milik perusahaan
- komunikasi data untuk keperluan administrasi antar instansi dan perangkat pemerintah daerah hingga tingkat kecamatan/kelurahan.

Hasil penelitian juga diharapkan menjadi langkah awal penelitian lanjutan tentang :

- perbaikan dan penyempurnaan *codec* RS yang ada
- pengembangan *codec* untuk blok yang lebih panjang
- implementasi jenis-jenis *codec* lainnya.

1.4. Metode Penelitian

Untuk mendapatkan hasil penelitian yang diinginkan, digunakan metode penelitian dengan langkah-langkah sebagai berikut :

1. Penentuan parameter kode RS :
   a) Mengamati m dan t kode RS yang sering digunakan.
   b) Mengamati spesifikasi FPGA yang digunakan.

c) Mengamati spesifikasi peralatan komunikasi radio dan modem yang digunakan.

d) Menentukan parameter kode RS berdasar tingkat kerumitan dan kemampuan peralatan.

2. Rancang bangun *codec* RS yang dilakukan dengan urutan langkah :

a) Merancang *codec* RS dengan parameter yang telah ditentukan

b) Memasukkan (*entry*) rancangan ke Xilink Foundation Tools

c) Melakukan langkah simulasi hasil rancangan

d) Melakukan langkah implementasi

e) Melakukan langkah pemrograman / *loading*

3. Pengujian *codec* RS, dengan proses sebagai berikut :

a) Menguji *encoder* secara mandiri

b) Menguji *decoder* secara mandiri

c) Membangkitkan sekumpulan data sebagai masukan *encoder*.

d) Membandingkan data keluaran *decoder* dengan masukan *encoder*, dalam kondisi tanpa kesalahan bit (simulasi kondisi tanpa *noise*).

e) Membangkitkan bit-bit *error* mulai dari 0 *error* hingga t+1 *error*, dan menyuntikkan ke keluaran *encoder*.

f) Membandingkan data keluaran *decoder* dengan masukan *encoder*, dari beberapa bit *error*.

g) Mencocokkan dengan codec RS *teoritis*

4. Evaluasi *codec* pada komunikasi data lewat kanal radio :

   a) Menguji unjuk kerja komunikasi data lewat kanal radio sebelum dipasang *codec* RS

   b) Menguji unjuk kerja komunikasi data lewat kanal radio setelah dipasang *codec* RS.

5. Pembuatan kesimpulan

6. Penyusunan laporan/tesis dengan susunan sebagai berikut :

   a) BAB I, berisi pendahuluan.

   b) BAB II, membahas teori-teori yang digunakan

   c) BAB III, membahas proses rancang bangun codec di FPGA

   d) BAB IV, membahas proses pengujian codec dan evaluasinya pada kanal radio

   e) BAB V, berisi penutup.

1.5. Kontribusi Penelitian

Sesuai dengan latar belakang dan tujuannya, kontribusi penelitian ini lebih banyak ditujukan untuk keperluan-keperluan yang bersifat aplikatif. Di antaranya adalah: pengaturan *plant*, komunikasi data antar PC untuk *e-government*. Selain itu, juga dapat digunakan sebagai referensi penelitian aplikatif lanjutan untuk peningkatan performa codec RS yang telah dirancang bangun, maupun rancang bangun tipe-tipe codec lain yang lebih unggul untuk aplikasi di lapangan.

BAB II
TEORI DASAR

2.1. Konsep Grup, Field, dan Field Biner

Kode Reed-Solomon (RS) adalah kode yang bekerja dalam areal aljabar *field*. Dalam pembentukannya, kode ini menggunakan tabel Galois field (GF). Galois field yang umum dipakai dalam implementasi adalah GF(2^m), di mana m adalah jumlah bit persimbol yang dikirim pada saat yang sama. Untuk itu, materi yang perlu diketahui lebih dahulu adalah : konsep grup, *field*, dan *field* biner GF(2).

2.1.1. Grup

Misalkan G adalah suatu set yang terdiri dari beberapa elemen. Jika operasi biner * pada G yang diterapkan untuk pasangan elemen a dan b ternyata menghasilkan elemen ketiga c = a * b (dan itu juga terjadi untuk seluruh elemen dalam G), maka G disebut tertutup dalam operasi *.

Sebuah set G dengan operasi biner * disebut grup jika memenuhi beberapa syarat berikut **(Lin,1983)**:

- Operasi biner * bersifat asosiatif.
- G mengandung elemen e, yang untuk sembarang elemen a dalam G berlaku: a * e = e * a = a. Elemen e disebut elemen identitas dari G.
- Untuk sembarang a dalam G, terdapat elemen a' dalam G yang mempunyai hubungan : a * a' = a' * a = e. Elemen a' disebut invers dari a, dan sebaliknya.

Sebuah grup G juga disebut komutatif, jika operasi biner untuk sembarang elemen a dan b dalam G memenuhi : a * b = b * a. Jumlah elemen dalam grup disebut orde dari grup. Sebuah grup dengan orde terbatas (*finite order*) disebut grup terbatas (*finite group*).

Untuk sembarang bilangan bulat positif m, dapat disusun grup dengan orde m dengan menggunakan operasi biner. Grup dengan orde m memiliki set G = {0, 1, 2, ..., m-1}. Jika + adalah operasi penjumlahan bilangan real, maka definisi operasi biner ⊞ pada G adalah sebagai berikut **(Lin,1983)**:

> *Untuk sembarang i dan j dalam G , i ⊞ j = r . Bilangan r adalah sisa dari pembagian i + j terhadap m dan merupakan bilangan bulat antara 0 dan m-1 (Algoritma pembagian Euclid's).*

Karena r terdapat dalam G, maka G tertutup untuk operasi biner ⊞. Operasi biner ini disebut dengan penjumlahan modulo m.

Elemen identitas dari operasi ini adalah 0. Invers dari i adalah m - i , dan sebaliknya. Sedang invers 0 adalah 0 itu sendiri. Karena penjumlahan bilangan real bersifat komutatif, maka penjumlahan modulo m juga komutatif. Selain itu, penjumlahan modulo m juga bersifat asosiatif. Tabel 2.1. menggambarkan penjumlahan modulo 5, yang mempunyai set grup G = {0, 1, 2, 3, 4}. Grup ini mempunyai orde sebesar 5.

Tabel 2.1 Penjumlahan modulo 5

⊞	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Selain *finite group* dengan operasi penjumlahan, dapat juga disusun *finite group* dengan operasi perkalian. Untuk bilangan prima p (p = 2, 3, 5, 7, ...), grup dengan orde p memiliki set G = {1, 2, ..., p-1}. Jika . adalah operasi perkalian bilangan real, maka definisi operasi biner ⊡ pada G adalah sebagai berikut **(Lin,1983)**:

> *Untuk sembarang i dan j dalam G , i ⊡ j = r . Bilangan r adalah sisa dari pembagian i . j terhadap p. Harus diingat bahwa i . j tidak terbagi habis oleh p. Elemen grup operasi penjumlahan adalah* $0 < r < p$.

Karena r terdapat dalam G, maka G tertutup untuk operasi biner ⊡. Operasi biner ini disebut dengan perkalian modulo p. Jika p bukan bilangan prima, maka set G = {1, 2, ..., p-1} bukan merupakan grup dalam perkalian modulo p.

Elemen identitas dari operasi ini adalah 1. Jika a terdapat dalam G dan memenuhi perkalian (modulo p) a ⊡ i = i ⊡ a = 1, maka a adalah invers i dan

sebaliknya. Perkalian modulo p ini bersifat asosiatif dan komutatif. Tabel 2.2. menggambarkan perkalian modulo 5 yang mempunyai elemen G = {1,2,3,4}.

Tabel 2.2 Perkalian modulo 5

⊡	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

2.1.2. Field

Sistem aljabar lain yang menggunakan konsep grup adalah *field*. Secara mudah dapat dikatakan bahwa *field* adalah suatu set elemen, yang kepadanya dapat dilakukan operasi penjumlahan, pengurangan, perkalian dan pembagian, tanpa meninggalkan set. Operasi penjumlahan dan perkaliannya memenuhi hukum komutatif, asosiatif dan distributif. Sedangkan definisi formal untuk *field* adalah sebagai berikut **(Lin,1983)**:

> *Misalkan F adalah suatu set elemen dengan operasi biner penjumlahan "+" dan perkalian "." .Set F bersama dengan dua operasi biner disebut field jika memenuhi syarat-syarat berikut :*
>
> - *F adalah grup komutatif pada operasi penjumlahan. Elemen identitas pada operasi biner penjumlahan adalah zero, dan dinotasikan dengan 0.*

- *Set dari elemen-elemen nonzero dalam F adalah grup komutatif pada operasi biner perkalian. Elemen identitas pada operasi perkalian disebut elemen unit, yang dinotasikan sebagai 1.*
- *Perkalian adalah distributif dari penjumlahan. Sehingga, jika ada tiga elemen a, b, dan c dalam F, maka :*

$$a . (b + c) = a . b + a . c$$

Jumlah elemen dalam *field* disebut orde *field.* Sebuah *field* dengan orde terbatas disebut *finite field.* Invers penjumlahan dari elemen *field* a, yang dinotasikan sebagai –a, adalah elemen field yang menghasilkan 0 jika dijumlahkan dengan a (a+(-a)= 0). Invers perkalian a, yang dinotasikan sebagai a^{-1}, adalah elemen field yang menghasilkan 1, jika dikalikan dengan a ($a . a^{-1} = 1$).

Untuk sembarang bilangan prima p, dapat dibentuk *finite field* yang mengandung p elemen. Untuk sembarang bilangan positif m, dimungkinkan untuk memperluas *prime field* GF(p) menjadi sebuah *field* dengan elemen-elemen p^m yang disebut sebagai *field* ekstensi dari GF(p), dan dinotasikan sebagai GF(p^m). Jadi orde dari sembarang *finite field* adalah pangkat dari bilangan prima. *Finite field* disebut juga sebagai *Galois Field*, untuk menghormati penemunya.

2.1.3. Field Biner (Binary Field)

Sebenarnya, kode dapat dibentuk dengan simbol-simbol dari sembarang GF(q) dengan q adalah bilangan prima p atau pangkat dari bilangan prima p. Tetapi pada transmisi data digital dan sistem penyimpanan/memori, paling banyak

digunakan kode dengan simbol dari GF(2) dan GF(2^m). Dalam aritmetika *field* biner, digunakan penjumlahan dan perkalian modulo biner.

Polinomial f(x) dengan satu variabel x dan koefisien-koefisien dari GF(2) mempunyai bentuk berikut :

$$f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_n x^n \qquad (2.1)$$

di mana $f_i = 0$ atau 1 untuk $0 \le i \le n$. Derajat polinomial adalah pangkat terbesar dari x yang koefisiennya bukan 0. Dua polinomial dari GF(2) dengan derajat 1 adalah x dan 1+x. Empat polinomial dari GF(2) dengan derajat 2 adalah : x^2 , $1+x^2$, $x+x^2$, $1+x+x^2$. Umumnya, terdapat 2^n polinomial dari GF(2) berderajat n.

Polinomial dari GF(2) dapat dijumlahkan, dikurangi, dikalikan, atau dibagi dengan cara biasa. Misalkan $g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_m x^m$ adalah polinomial lain dari GF(2), maka penjumlahan f(x) dan g(x) dilakukan dengan menjumlahkan koefisien yang pangkat x-nya sama. Jika diasumsikan $m \le n$, maka :

$$f(x) + g(x) = (f_0 + g_0) + (f_1 + g_1) x + \ldots + (f_m + g_m) x^m + \ldots + f_n x^n \qquad (2.2)$$

di mana $f_i + g_i$ dalam penjumlahan modulo 2.

Perkalian f(x) dan g(x) sedikit lebih rumit dan menghasilkan persamaan yang lebih panjang, yaitu :

$$f(x) \cdot g(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n+m} x^{n+m} \qquad (2.3)$$

di mana :

$c_0 = f_0 g_0$

$c_1 = f_0 g_1 + f_1 g_0$

$c_2 = f_0 g_2 + f_1 g_1 + f_2 g_0$

...

$c_i = f_0 g_i + f_1 g_{i-1} + f_2 g_{i-2} + \ldots + f_i g_0$

...

$c_{n+m} = f_n g_m$

Perkalian dan penjumlahan merupakan operasi modulo 2. Polinomial dari GF(2) memenuhi sifat-sifat komutatif, asosiatif dan distributif.

Untuk derajat g(x) bukan 0, maka pembagian f(x) terhadap g(x) akan menghasilkan pasangan polinomial unik q(x) yang disebut *quotient* (hasil pembagian) dan r(x) yang disebut *remainder* (sisa pembagian). Hubungan antara keempat polinomial ditulis dalam persamaan :

$$f(x) = g(x) \,.\, q(x) + r(x) \qquad (2.4)$$

di mana derajat r(x) lebih kecil daripada g(x). Persamaan ini dikenal sebagai algoritma pembagian Euclid (*Euclid's Algorithm Division*). Jika sisa pembagian f(x) terhadap g(x) adalah 0 (r(x) = 0), maka g(x) adalah faktor dari f(x). Untuk bilangan real, a adalah akar polinomial f(x) jika f(x) dapat dibagi (habis) dengan x-a.

Polinomial p(x) dari GF(2) dengan derajat m disebut *irreducible* dari GF(2), jika tidak dapat dibagi (habis) oleh sembarang polinomial dari GF(2) yang derajatnya lebih kecil dari m tapi lebih besar dari nol. Polinomial *irreducible* p(x) dengan derajat m disebut primitif jika bilangan bulat positif terkecil n, yang polinomialnya dapat membagi habis $x^n + 1$ adalah $n = 2^m - 1$. Daftar beberapa polinomial primitif untuk beberapa nilai m ditunjukkan dalam tabel 2.3. Untuk m yang ada dalam tabel, polinomial primitifnya bisa lebih dari satu.

Tabel 2.3 Daftar polinomial primitif untuk beberapa nilai m

m	Polinomial Primitif	m	Polinomial Primitif
3	$1 + x + x^3$	14	$1 + x + x^6 + x^{10} + x^{14}$
4	$1 + x + x^4$	15	$1 + x + x^{15}$
5	$1 + x^2 + x^5$	16	$1 + x + x^3 + x^{12} + x^{16}$
6	$1 + x + x^6$	17	$1 + x^3 + x^{17}$
7	$1 + x^3 + x^7$	18	$1 + x^7 + x^{18}$
8	$1 + x^2 + x^3 + x^4 + x^8$	19	$1 + x + x^2 + x^5 + x^{19}$
9	$1 + x^4 + x^9$	20	$1 + x^3 + x^{20}$
10	$1 + x^3 + x^{10}$	21	$1 + x^2 + x^{21}$
11	$1 + x^2 + x^{11}$	22	$1 + x + x^{22}$
12	$1 + x + x^4 + x^6 + x^{12}$	23	$1 + x^5 + x^{23}$
13	$1 + x + x^3 + x^4 + x^{13}$	24	$1 + x + x^2 + x^7 + x^{24}$

Pembentukan Elemen GF(2^m)

Elemen-elemen Galois Field 2^m (untuk $m > 1$) yang berjumlah 2^m dapat dibentuk dari *field* biner GF(2). Modal awal pembentukan elemen GF(2^m) adalah : elemen 0 dan 1 dari GF(2) serta elemen baru yang disebut α. Dengan menggunakan perkalian "." , dapat dibentuk urutan pangkat α berikut :

$0 . 0 = 0$

$0 . 1 = 1 . 0 = 0$

$1 . 1 = 1$

$0 . \alpha = \alpha . 0 = 0$

$1 . \alpha = \alpha . 1 = \alpha$

$\alpha^2 = \alpha \cdot \alpha$

$\alpha^3 = \alpha \cdot \alpha \cdot \alpha$

....

$\alpha^j = \alpha \cdot \alpha \cdot \alpha \ldots \alpha$ (j kali)

....

Dan seterusnya (2.5)

Dengan mengikuti perkalian di atas, dapat dituliskan persamaan-persamaan berikut :

$0 \cdot \alpha^j = \alpha^j \cdot 0 = 0$

$1 \cdot \alpha^j = \alpha^j \cdot 1 = \alpha^j$

$\alpha^i \cdot \alpha^j = \alpha^j \cdot \alpha^i = \alpha^{i+j}$ (2.6)

Sehingga set elemen untuk operasi perkalian "." Adalah $F = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^j, \ldots\}$. Elemen 1 sering juga ditulis dengan α^0.

Operasi penjumlahan "+" dalam GF(2) menyatakan bahwa $0 + 0 = 0$, $1 + 1 = 0$, dan $0 + 1 = 1 + 0 = 0$. Hal ini juga berlaku untuk elemen α^i. Sehingga diperoleh :

$0 + \alpha^i = \alpha^i + 0 = \alpha^i$

$\alpha^i + \alpha^i = 0$ (2.7)

Operasi perkalian dan penjumlahan di atas digunakan untuk membentuk elemen $GF(2^m)$.

Untuk $m = 4$, polinomial primitifnya dapat menggunakan $p(x) = 1 + x + x^4$. Polinomial $p(x)$ tersebut merupakan polinomial primitif dalam GF(2). Jika α di substitusikan dalam polinomial $p(x) = 0$, akan diperoleh $p(\alpha) = 1 + \alpha + \alpha^4 = 0$, sehingga: $\alpha^4 = 1 + \alpha$. Elemen identitas $\alpha^4 = 1 + \alpha$ ini dapat digunakan berulang-ulang untuk membentuk elemen lain. Contoh :

$\alpha^5 = \alpha \cdot \alpha^4 = \alpha (1 + \alpha) = \alpha + \alpha^2$

$\alpha^6 = \alpha \cdot \alpha^5 = \alpha (\alpha + \alpha^2) = \alpha^2 + \alpha^3$

$\alpha^7 = \alpha \cdot \alpha^6 = \alpha (\alpha^2 + \alpha^3) = \alpha^3 + \alpha^4 = \alpha^3 + 1 + \alpha = 1 + \alpha + \alpha^3$

Daftar elemen GF(2^4) yang dibentuk dari polinomial $p(x) = 1 + x + x^4$ ditunjukkan oleh tabel 2.4. Elemen-elemen tersebut dapat ditulis dalam tiga representasi, yaitu bentuk : pangkat, polinomial dan 4 *tuple*.

Tabel 2.4 Elemen GF(2^4) yang dibangkitkan oleh $p(x) = 1 + x + x^4$

Bentuk Pangkat	Bentuk Polinomial	Bentuk 4 tuple
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

LSB MSB

Dalam $GF(2^4)$, hasil perkalian elemen α^i dengan α^j diperoleh dengan cara menjumlahkan eksponennya, dan menggunakan fakta $\alpha^{15} = 1$. Fakta ini juga berlaku untuk eksponen-eksponen kelipatan 15. Sebagai contoh : $\alpha^{12} . \alpha^7 = \alpha^{19} = \alpha^4$.

Sementara itu, hasil pembagian elemen α^j terhadap α^i diperoleh dengan cara mengalikan elemen α^j terhadap invers dari α^i. Invers dari α^i adalah α^{15-i}. Sebagai contoh : $\alpha^4 / \alpha^{12} = \alpha^4 . \alpha^3 = \alpha^7$.

Penjumlahan α^i terhadap α^j diperoleh dengan menggunakan bentuk polinomial dalam tabel 2.4. Sebagai contoh : $\alpha^5 + \alpha^7 = (\alpha + \alpha^2) + (1 + \alpha + \alpha^3) = 1 + \alpha^2 + \alpha^3 = \alpha^{13}$.

2.2.1. Implementasi Operasi Penjumlahan Dan Pengurangan Dalam $GF(2^4)$

Rangkaian *codec* RS banyak menggunakan operasi penjumlahan baik di sisi *encoder* maupun *decoder*. Operasi penjumlahan dua elemen field dalam $GF(2^4)$ dilakukan dengan menggunakan kaidah dalam persamaan 2.7.

Untuk $GF(2^4)$, α adalah elemen primitif dari polinomial minimal $\phi(x) = 1 + x + x^4$. Misalkan suatu elemen *field* β akan dijumlahkan dengan elemen *field* γ , maka elemen β dapat dinyatakan dalam bentuk polinomial α berikut **(Lin,1983)**:

$$\beta = b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3 \qquad (2.8)$$

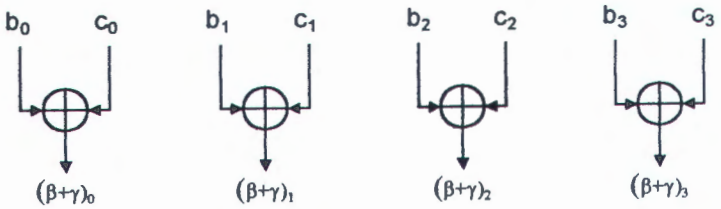
Sedangkan elemen γ dinyatakan dalam bentuk polinomial :

$$\gamma = c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 \qquad (2.9)$$

Penjumlahan β dengan γ dilakukan dengan menjumlahkan koefisien β dengan γ yang mempunyai pangkat α sama. Hasilnya adalah :

$$\beta + \gamma = (b_0 + c_0) + (b_1 + c_1) \alpha + (b_2 + c_2) \alpha^2 + (b_3 + c_3) \alpha^3 \qquad (2.10)$$

Rangkaian penjumlahannya ditunjukkan dalam gambar 2.1.



Gambar 2.1. Implementasi penjumlahan elemen *field* β dengan γ

Operasi pengurangan dalam GF(2^4) sama sederhananya dengan operasi penjumlahan dalam GF(2^4). Karena GF(2^4) adalah field dengan karakteristik 2, maka operasi penjumlahan identik dengan operasi pengurangan **(Michelson, 1985)**. Ini berarti tanda minus dapat langsung diganti dengan plus, dan rangkaiannya serupa dengan rangkaian penjumlahan.

2.2.2. Implementasi Operasi Perkalian Dan Pembagian Dalam GF(2^4)

Operasi perkalian dalam GF(2^m) juga banyak ditemukan pada *codec* RS . Perkalian dapat dilakukan antara suatu elemen *field* dengan elemen tetap atau elemen tidak tetap/variabel dalam GF(2^4).

Jika elemen β dikalikan dengan elemen tetap α, maka kedua sisi persamaan di atas harus dikalikan dengan α, sehingga bentuk persamaannya menjadi **(Lin,1983)**:

$$\beta\,\alpha = \alpha\,(b_0 + b_1\,\alpha + b_2\,\alpha^2 + b_3\,\alpha^3)$$

$$= b_0\,\alpha + b_1\,\alpha\,\alpha + b_2\,\alpha^2\,\alpha + b_3\,\alpha^3\,\alpha$$

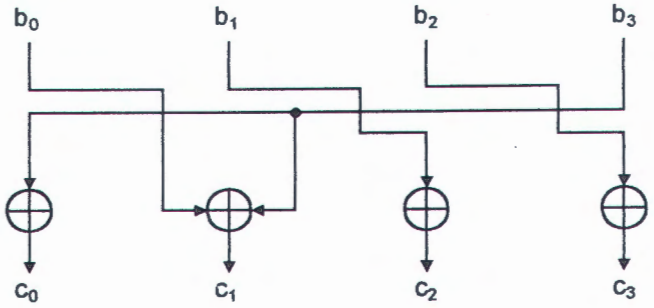
$$= b_0\,\alpha + b_1\,\alpha^2 + b_2\,\alpha^3 + b_3\,\alpha^4 \qquad (2.11)$$

Dari tabel 2.4 diperoleh bahwa $\alpha^4 = 1 + \alpha$, maka persamaan di atas menjadi :

$$\beta\,\alpha = b_0\,\alpha + b_1\,\alpha^2 + b_2\,\alpha^3 + b_3\,(1 + \alpha)$$

$$= b_3 + (b_0 + b_3)\,\alpha + b_1\,\alpha^2 + b_2\,\alpha^3 \qquad (2.12)$$

Persamaan tersebut dapat diimplementasikan dalam rangkaian pada gambar 2.2.



Gambar 2.2. Implementasi perkalian elemen *field* β dengan α

Implementasi perkalian dengan elemen *field* lain dilakukan dengan cara yang serupa dengan di atas.

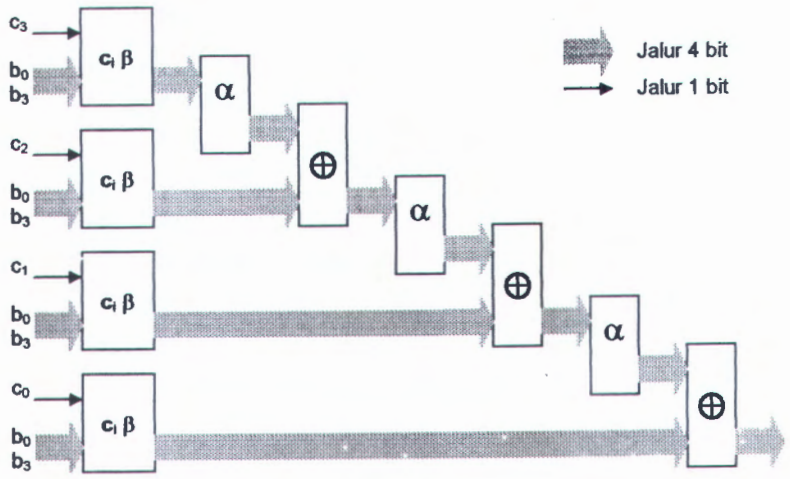
Perkalian elemen β dengan elemen γ yang variabel tidak sederhana seperti perkalian dengan elemen tetap. Jika β dan γ adalah elemen-elemen field dengan bentuk polinomial dalam persamaan 2.8 dan 2.9, maka perkaliannya menghasilkan persamaan **(Lin,1983)**:

$$\beta\,\gamma = (((c_3\,\beta)\,\alpha + c_2\,\beta)\,\alpha + c_1\,\beta)\,\alpha + c_0\,\beta \qquad (2.13)$$

Proses yang dilakukan pada operasi perkalian di atas mempunyai urutan langkah berikut **(Lin,1983)**:

1. Kalikan $c_3\,\beta$ dengan α dan jumlahkan hasilnya dengan $c_2\,\beta$
2. Kalikan $(c_3\,\beta)\,\alpha + c_2\,\beta$ dengan α dan jumlahkan hasilnya dengan $c_1\,\beta$
3. Kalikan $((c_3\,\beta)\,\alpha + c_2\,\beta)\,\alpha + c_1\,\beta$ dengan α dan jumlahkan dengan $c_0\,\beta$

Implementasi rangkaian untuk persamaan 2.13 membutuhkan 3 blok pengali $\beta\ \alpha$ (pada gambar 2.2), 3 blok penjumlah 4 bit (pada gambar 2.1) dan 3 blok operasi c_i AND β. Diagram blok rangkaian pengali elemen *field* β dengan γ ditunjukkan pada gambar 2.3.



Gambar 2.3. Implementasi perkalian elemen *field* β dengan γ

Selain diimplementasikan dengan cara di atas, perkalian dua elemen *field* β dengan γ dapat diperoleh dengan menggunakan persamaan berikut : **(Gill, 2002)** :

$$a_0 = b_0 c_0 + b_1 c_3 + b_2 c_2 + b_3 c_1$$
$$a_1 = b_0 c_1 + b_1 c_0 + b_1 c_3 + b_2 c_2 + b_2 c_3 + b_3 c_1 + b_3 c_2$$
$$a_2 = b_0 c_2 + b_1 c_1 + b_2 c_0 + b_2 c_3 + b_3 c_2 + b_3 c_3$$
$$a_3 = b_0 c_3 + b_1 c_2 + b_2 c_1 + b_3 c_0 + b_3 c_3 \qquad (2.14)$$

Pada implementasinya, perkalian koefisien dilakukan dengan menggunakan gerbang AND, sedang penjumlahan dengan gerbang EXOR.

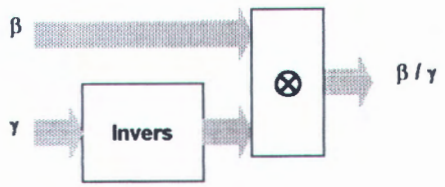
Untuk operasi pembagian, hasil pembagian elemen *field* β (dalam α^j) terhadap elemen γ (dalam α^i) diperoleh dengan mengalikan elemen α^j terhadap invers dari α^i (di mana invers dari α^i adalah $\alpha^{15 - i}$). Implementasi blok invers

diperoleh dengan memanfaatkan tabel kebenaran (*thruth table*) yang ditunjukkan pada tabel 2.5.

Tabel 2.5. Thruth table untuk operasi invers

Input				Output / Invers			
I_0	I_1	I_2	I_3	O_0	O_1	O_2	O_3
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	0	1	1
0	0	1	1	0	1	0	1
0	1	0	0	1	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	1	1	1	0
0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	0	1	0
1	1	0	0	0	1	1	1
1	1	0	1	1	0	1	0
1	1	1	0	0	1	1	0
1	1	1	1	0	0	0	1

Berdasar tabel 2.5 dapat dibuat rangkaian logika kombinasi dengan 4 masukan dan 4 keluaran. Dengan demikian, dapat dibuat rangkaian pembagi yang diagram bloknya ditunjukan dalam gambar 2.4. Semua masukan dan keluaran mempunyai lebar 4 bit.



Gambar 2.4. Implementasi pembagian elemen *field* β dengan γ

2.3. Kode Reed-Solomon

Kode Reed Solomon (RS) adalah sub kelas dari kode BCH (Bose, Chauduri, dan Hocquenghem). Kelas kode BCH dikenal mempunyai "*large class of powerful random error-correcting cyclic codes*". Kode BCH terbagi atas sub kelas biner dan non biner. Kode RS merupakan sub kelas non biner.

Jika kode BCH biner menggunakan GF(2) untuk membentuk koefisien-koefisien polinomialnya, kode BCH non biner menggunakan GF(q). Karena itu, kode non biner disebut kode *q-ary* **(Lin,1983)**. Konsep dan sifat kode non biner serupa dengan kode biner, tetapi dengan sedikit modifikasi. Untuk sembarang bilangan bulat positif s dan t, kode BCH *q-ary* mempunyai panjang $n=q^s - 1$ dengan kemampuan memperbaiki *error* sebanyak t.

Kode RS merupakan sub kelas spesial kode BCH *q-ary* dengan s = 1. Kode RS dengan simbol-simbol kode dari GF(q) dan kemampuan koreksi sebesar t *error* mempunyai parameter-parameter (semuanya dalam simbol) :

Panjang blok	$n = q - 1$
Jumlah digit *parity check*	$n-k = 2t$
Jarak minimum	$d_{min} = 2t + 1$

di mana k adalah jumlah digit simbol informasi yang masuk. Untuk komunikasi digital, nilai q yang digunakan adalah 2^m, di mana m adalah jumlah bit dalam satu simbol.

Kode RS termasuk kode blok yang bekerja per-simbol. Berbeda dengan kode BCH biner yang memproses kode secara bit per bit, kode RS memproses kode secara simbol per simbol. Untuk tiap simbol yang dinyatakan dalam m bit, jumlah bit yang diproses secara simultan / paralel adalah m bit.

Jika data yang masuk dan keluar *codec* berupa aliran serial (perbit), maka pada *codec* perlu ditambahkan rangkaian konverter serial ke paralel dan paralel ke serial **(Chio, ---)**. Tetapi jika data yang masuk dan keluar *codec* berupa aliran paralel (persimbol), rangkaian konverter tidak perlu ditambahkan. Untuk keluaran *encoder* berupa aliran paralel, modulasi yang digunakan adalah modulasi *q-ary*. Jika berupa aliran serial, modulasi yang digunakan adalah modulasi biner. Pengiriman kode dengan bentuk aliran serial, menyerupai metode *interlaced*, sehingga membuat kode RS mampu mengatasi *error burst*.

2.4. Encoding Kode RS

Jika α adalah elemen primitif dalam $GF(2^m)$, maka generator polinomial dari kode RS primitif dengan kemampuan koreksi t *error* untuk kode sepanjang $2^m - 1$ adalah :

$$g(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^{2t})$$

$$= g_0 + g_1 x + g_2 x^2 + \ldots + g_{2t-1} x^{2t-1} + x^{2t} \qquad (2.15)$$

Polinomial generator g(x) memiliki α, α^2, .. α^{2t} sebagai akar-akarnya, dan mempunyai koefisien-koefisien (g_0 , g_1, ... , g_{2t-1}) yang diambil dari $GF(2^m)$. Kode yang dibangkitkan oleh g(x) adalah kode *cyclic* (n, n-2t).

Misalkan u(x) adalah polinomial informasi yang akan dikodekan, dan mempunyai persamaan :

$$u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{k-1} x^{k-1} \qquad (2.16)$$

maka untuk bentuk sistematik, digit *parity check* sebanyak t simbol mempunyai koefisien dari sisa pembagian x^{2t} u(x) terhadap g(x). Polinomial sisa pembagiannya adalah :

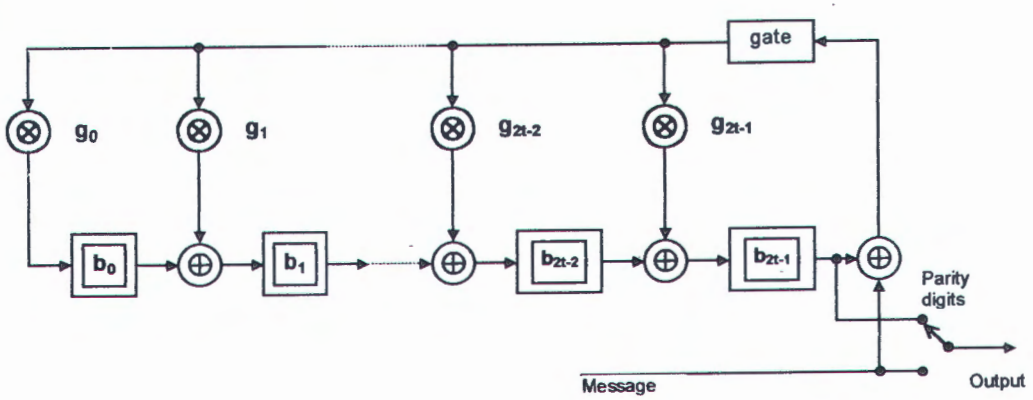
$$b(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{2t-1} x^{2t-1} \qquad (2.17)$$

Codeword yang dihasilkan adalah gabungan dari informasi yang masuk dengan polinomial sisa pembagian tersebut.

Secara teoritis, langkah-langkah yang dilakukan untuk mendapatkan codeword bentuk sistematik adalah sebagai berikut **(Lin,1983)**:

1. Mengalikan informasi u(x) dengan x^{n-k}
2. Mendapatkan sisa pembagian b(x) yang diperoleh dari pembagian x^{n-k} u(x) terhadap g(x)
3. Menggabungkan b(x) dan x^{n-k} u(x), sehingga menjadi *codeword* dengan polinomial $b(x) + x^{n-k} u(x)$

Untuk implementasi *hardware*, rangkaian *encoder* kode RS sistematik dapat disusun dari *Linier Feedback Shift Register* (LSFR) yang mempunyai diagram blok dalam gambar 2.5 **(Lin,1983)**.



Gambar 2.5 Diagram blok *encoder* kode RS sistematik

Tiap register menyimpan m simbol, sedangkan perkalian dengan koefisien generator merupakan perkalian dalam $GF(2^m)$.

Ketika sedang menerima informasi sebanyak k simbol, saklar pada keluaran dihubungkan dengan masukan, sehingga digit informasi langsung terkirim keluar *encoder*. Pada saat yang sama, *encoder* juga mengaktifkan *gate* agar di register b_0 hingga b_{2t-1} tersimpan digit paritas yang dihasilkan oleh masuknya tiap digit informasi.

Saat k simbol informasi telah diterima, *gate* segera dinonaktifkan, dan saklar keluaran dipindah ke digit paritas. Kemudian, n-k simbol paritas terakhir yang tersimpan dikirim keluar untuk melengkapi *codeword*-nya.

2.5. Decoding Kode RS

Misalkan polinomial *codeword* yang dikirim mempunyai persamaan sebagai berikut :

$$v(x) = v_0 + v_1 x + v_2 x^2 + \ldots + v_{n-1} x^{n-1} \qquad (2.18)$$

sedangkan polinomial kode yang diterima mempunyai persamaan sebagai berikut :

$$r(x) = r_0 + r_1 x + r_2 x^2 + \ldots + r_{n-1} x^{n-1} \qquad (2.19)$$

maka polinomial *error* yang diakibatkan oleh kanal adalah selisih antara v(x) dengan r(x) dan mempunyai persamaan :

$$e(x) = r(x) - v(x) = e_0 + e_1 x + e_2 x^2 + \ldots + e_{n-1} x^{n-1} \qquad (2.20)$$

di mana $e_i = r_i - v_i$ adalah koefisien *error* yang simbolnya diperoleh dari $GF(2^m)$

Nilai sindrom yang diakibatkan oleh adanya *error* dapat diperoleh dengan mengaplikasikan lagi aturan *encoding* pada *word* yang diterima. Hal itu dilakukan dengan mengevaluasi r(x) pada akar-akar generator polinomial yang digunakan, sehingga diperoleh :

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

$$= v(\alpha^i) + e(\alpha^i)$$

$$= e(\alpha^i) \qquad (2.21)$$

di mana $1 \le i \le 2t$. Pada persamaan tersebut $v(\alpha^i) = 0$, karena $\alpha^1, \alpha^2, \ldots, \alpha^{2t}$ adalah akar dari tiap *codeword* v(x) yang dikirim (Catatan: tiap *codeword* merupakan perkalian dengan polinomial generator g(x)). Tampak bahwa 2t buah sindrom parsial S_i hanya tergantung pada pola *error* e(x) dan tidak tergantung pada *codeword* tertentu r(x) yang diterima.

Polinomial *error* e(x) mempunyai koefisien-koefisien *non zero* yang menyatakan lokasi *error*. Maka jika terjadi t *error* pada *word* yang diterima, nilai sindrom dapat ditulis sebagai :

$$S_i = \sum_{j=1}^{t} Y_j X_j^i \qquad i = 1, 2, \ldots, 2t \qquad (2.22)$$

di mana X_j adalah *error locator* dari *error* ke j dan Y_j adalah *error value*. Maka dengan mendapatkan sindrom S_i dapat dicari X dan Y-nya.

Secara umum langkah-langkah yang dilakukan dalam proses *decoding* kode RS adalah sebagai berikut **(Michelson, 1985)** :

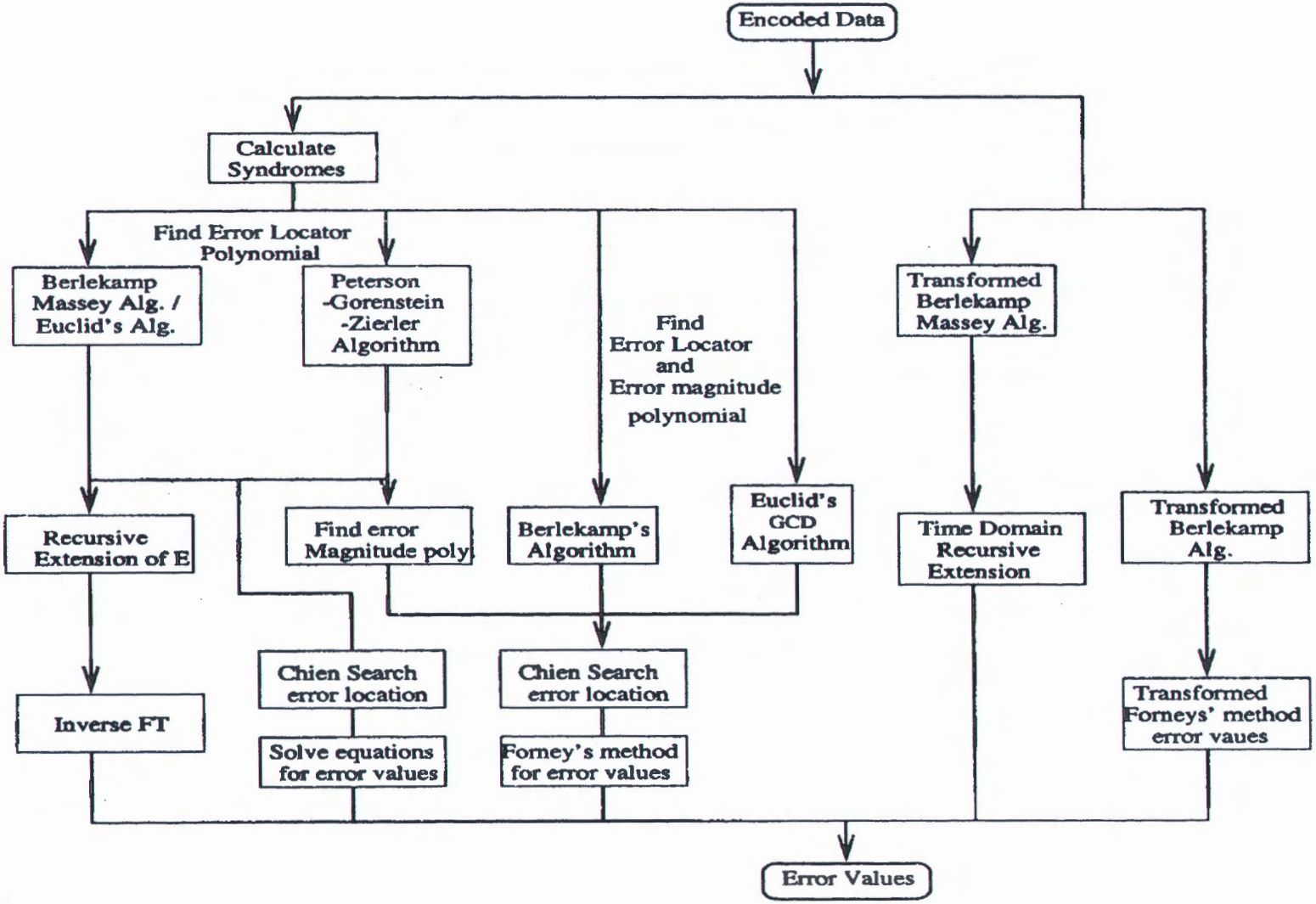
1. Menghitung nilai sindrom S_i, $i = 1, 2, \ldots, 2t$
2. Menentukan polinomial *error-locator* $\sigma(x)$ dengan menggunakan nilai sindrom yang diperoleh
3. Mencari akar dari $\sigma(x)$, yang merupakan *error locator*.
4. Menghitung *error value* berdasar *error locator* yang diperoleh
5. Memperbaiki *error* yang terindikasi.

Di antara kelima langkah tersebut, langkah yang paling kompleks adalah penentuan polinomial *error-locator* $\sigma(x)$. Beberapa metode yang dipakai dalam langkah ini antara lain adalah : algoritma Massey-Berlekamp, metode Peterson dan algoritma Euclidian.

Walaupun secara umum *decoding* kode RS dilakukan dengan lima langkah di atas, namun implementasinya cukup beragam. Keragaman teknik decoding kode RS ditunjukkan oleh diagram alir dalam gambar 2.6. Sebagai contoh, algoritma Berlekamp maupun Euclid dapat digunakan untuk mencari *error locator polynomial* saja atau sekaligus untuk mencari *error magnitude polynomial*. Pilihan tersebut menghasilkan langkah lanjutan yang berbeda.

Jika algoritma Berlekamp atau Euclid digunakan untuk mendapatkan *error locator polynomial* dan *error magnitude polynomial*, maka tinggal dilakukan proses pencarian lokasi *error* dan nilai *error magnitude*. Proses pencarian dilakukan dengan menggunakan *Chien Search*. Selanjutnya, algoritma/metode Forney digunakan untuk mendapatkan *error value*. *Error value* inilah yang nantinya dijumlahkan dengan simbol yang mengalami *error* untuk mendapatkan simbol aslinya.

Metode-metode *decoding* kode RS tampaknya cukup berkembang. Karena itu, munculah metode-metode lain seperti : *time domain recursive*, *frequency domain*, *systolic array*, dan lain-lain. Kemunculan metode-metode ini tentunya bertujuan untuk mendapatkan sistem yang efisien dan sesuai dengan keperluan atau bentuk implementasi dari *decoder*.



Gambar 2.6 Diagram teknik decoding RS

2.5.1. Perhitungan Sindrom

Pada kode RS, 2t buah komponen sindrom S_i didefinisikan sebagai sisa pembagian *codeword* r(x) terhadap $x + \alpha^i$ **(Rappaport, 1996)**. Jadi jika *codeword* r(x) ditulis sebagai persamaan **(Lin, 1983)** :

$$r(x) = c_i(x)(x + \alpha^i) + b_i \qquad , i = 1, 2, \ldots, 2t \qquad (2.23)$$

di mana c_i adalah hasil pembagian dan b_i adalah sisa pembagian, maka sindrom S_i mempunyai persamaan :

$$S_i = b_i \qquad (2.24)$$

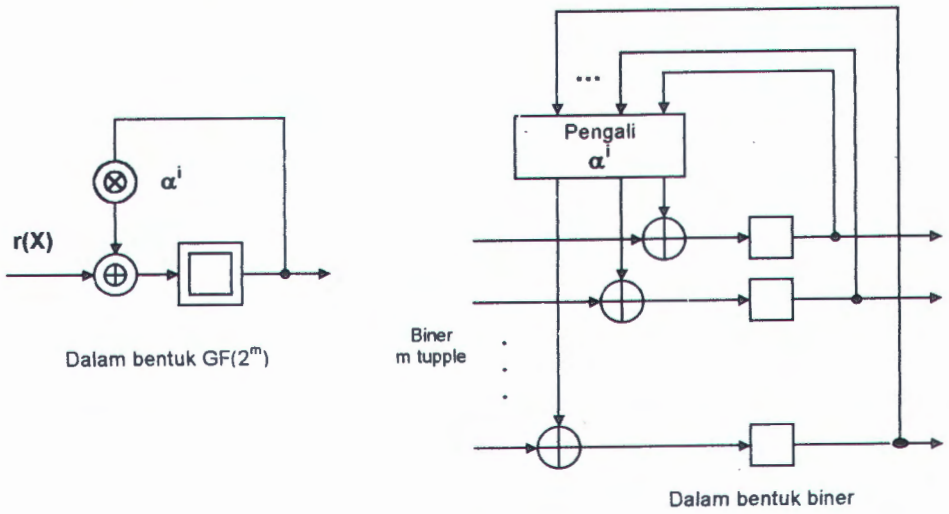
Pembuktian lain juga dapat dilakukan dengan mensubstitusi α^i pada tiap x pada persamaan r(x). Hasilnya adalah :

$$S_i = r(\alpha^i) = c_i(\alpha^i)(\alpha^i + \alpha^i) + b_i \qquad , i = 1, 2, \ldots, 2t$$

$$S_i = r(\alpha^i) = c_i(\alpha^i) \cdot 0 + b_i$$

$$S_i = r(\alpha^i) = b_i \qquad (2.25)$$

Perhitungan tersebut dapat diimplementasikan menjadi rangkaian *syndrome calculator* dengan diagram blok dalam gambar 2.7 **(Lin,1983)**.



Gambar 2.7 Diagram blok syndrome calculator

2.5.2. Peterson's Direct Solution Method

Peterson's direct solution method awalnya digunakan untuk kode BCH, kemudian disempurnakan oleh Gorenstein dan Zierler untuk kode Reed Solomon. Metode ini digunakan untuk menghitung koefisien polinomial *error locator* $\sigma(x)$. Satu set persamaan sindrom non linier simultan (dalam X) pada persamaan (2.22) dapat diubah menjadi satu set persamaan linier yang dipecahkan melalui konjungsi dengan $\sigma(x)$ **(Michelson, 1985)**. Hasil perubahannya adalah sebagai berikut :

$$S_{t+k} + \sigma_1 S_{t+k-1} + \ldots + \sigma_t S_k = 0 \qquad \text{untuk seluruh k} \qquad (2.26)$$

Di mana σ adalah koefisien-koefisien dari polinomial *error locator* $\sigma(x)$ yang mempunyai persamaan :

$$\sigma(x) = x^t + \sigma_1 x^{t-1} + \ldots + \sigma_t \qquad (2.27)$$

Persamaan (2.27) disebut sebagai persamaan *Newton's identities*.

Untuk kode RS yang mampu mengkoreksi t *error*, sindrom yang dihitung sebanyak 2t dengan nilai sindrom $S_1, S_2, \ldots, S_{2t}$. Dari nilai-nilai sindrom dapat disusun t buah persamaan simultan *Newton's identities* dengan nilai k dari 1 hingga t. Untuk t = 2, persamaan simultannya adalah sebagai berikut :

$$S_1 \sigma_2 + S_2 \sigma_1 = - S_3$$

$$S_2 \sigma_2 + S_3 \sigma_1 = - S_4 \qquad (2.28)$$

Kedua persamaan simultan tersebut harus diselesaikan untuk mendapat koefisien polinomial *error locator* σ_1 dan σ_2.

Jumlah persamaan yang digunakan untuk mendapatkan koefisien polinomial *error locator* sama dengan jumlah *error* aktual yang terjadi pada kode blok yang diterima. Untuk menentukan berapa jumlah persamaan yang digunakan, dilakukan

pemeriksaan terhadap determinan persamaan tiap kemungkinan *error* yang terjadi. Proses ini merupakan bagian dari proses *decoding*. Jumlah kemungkinan *error*nya disesuaikan dengan jumlah *error* maksimal yang mampu dikoreksi. Untuk t = 2, ada dua kemungkinan *error* yang terjadi (selain *no error*) adalah satu *error* dan dua *error*. Persamaaan simultan untuk satu *error* dan dua *error* masing-masing adalah :

$$[S_1][\sigma_1] = [-S_2]$$
$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} \qquad (2.29)$$

Determinan untuk dua *error* D_2 adalah $S_1 S_3 - S_2^2$. Dengan memeriksa determinan-determinan yang ada, dapat diperoleh jumlah *error* aktual yang terjadi . Jika $D_2 = 0$, maka kemungkinan terjadi satu *error*.

Peterson's direct method tidak efisien digunakan pada jumlah *error* yang banyak, karena memerlukan operasi perkalian, penjumlahan dan pembagian beberapa kali lipat. Pada kode BCH biner, metode Peterson sebaiknya digunakan untuk mengkoreksi maksimal 6 *error*. Jumlah *error* yang lebih besar akan membutuhkan operasi perkalian yang jumlahnya mendekati kuadrat dari jumlah *error* t yang mampu diperbaiki **(Michelson, 1985)**.

Untuk kode RS dengan t = 2, decodernya dapat diimplementasikan seperti berikut ini. Misalkan *word* yang diterima mempunyai polinomial r(x), dan simbol paritas diletakkan pada pangkat yang rendah, maka langkah-langkah proses *decoding* dengan metode *direct solution Peterson* adalah sebagai berikut **(Michelson, 1985)**:

1. Menghitung nilai sindrom S_i untuk $1 \leq i \leq 4$, di mana :

   $S_i = r(\alpha^i)$

2. Menentukan jumlah *error* yang terjadi pada *word* yang diterima.

   a. jika $S_i = 0$, $1 \leq i \leq 4$, maka *word* yang diterima adalah *codeword*, sehingga tidak perlu dilakukan proses koreksi

   b. Jika $D_2 = S_1 S_3 + S_2^2 \neq 0$, maka dianggap terjadi dua *error*

   c. Jika $D_2 = 0$ dan $S_1 \neq 0$, maka dianggap terjadi satu *error*.

3. Menghitung koefisien polinomial *error locator* :

   a. jika terjadi dua *error*, maka yang harus dihitung adalah :

   $\sigma_1 = (S_1 S_4 + S_2 S_3) / D_2$

   $\sigma_2 = (S_2 S_4 + S_3^2) / D_2$

   b. jika terjadi satu *error*, maka yang harus dihitung adalah :

   $\sigma_1 = X_1 = S_2 / S_1$

4. Jika terindikasi dua *error*, dengan menggunakan *Chien search* dilakukan penghitungan akar polinomial $\sigma(x) = x^2 + \sigma_1 x + \sigma_2$. Hasilnya harus berupa dua akar polinomial. Jika jumlah akar yang benar tidak didapatkan, maka dikeluarkan tanda deteksi *error*.

5. Setelah diperoleh *error locator* yang identik dengan akar polinomial, *error value* dapat dihitung dengan persamaan sindrom berikut :

   a. jika terjadi dua *error*, maka *error value*-nya adalah :

   $Y_1 = (S_1 X_2 + S_2) / (X_1 X_2 + X_1^2)$

   $Y_2 = (S_1 X_1 + S_2) / (X_1 X_2 + X_2^2)$

   b. jika terjadi satu *error*, maka *error value*-nya adalah :

   $Y_1 = S_1^2 / S_2$

6. Memperbaiki *word* yang diterima dengan cara menjumlahkan dengan *error value* hasil perhitungan di lokasi yang ditunjuk *error locator*.
7. Menghitung sindrom dari *word* yang sudah dikoreksi. Jika hasilnya tidak sama dengan 0, dikeluarkan tanda deteksi *error*.

Karena kode RS mempunyai *field* simbol kode dan *field locator* yang sama, maka seluruh perhitungan dalam proses *decoding* dilakukan dalam field $GF(2^m)$ dengan m yang sama. Pada field $GF(2^m)$, penjumlahan dan pengurangan merupakan operasi yang identik. Artinya, tanda minus pada koefisien polinomial $\sigma(x)$ dan pada perhitungan *error value* dapat diganti tanda plus. Selain itu, 2^m elemen field dapat dinyatakan sebagai bentuk biner m *tuple*. Penjumlahan diimplementasikan dalam penjumlahan modulo 2 yang dilakukan bit per bit.

2.5.3. Chien Search

Chien search adalah algoritma yang digunakan untuk memperoleh akar polinomial *error locator* $\sigma(x)$. Secara sistematis, $\alpha, \alpha^2, \ldots, \alpha^{n-1}$ di subsitusikan ke tiap x dalam polinomial $\sigma(x)$ untuk memperoleh nilai akar, jika substitusinya menghasilkan $\sigma(x) = 0$.

Operasi *Chien search* berdasar pada persamaan *Newton's identities* (2.27). Jika sisi kiri dan kanan persamaan (2.27) dibagi dengan x^t, akan diperoleh persamaan **(Michelson, 1985)**:

$$\sigma(x)/x^t = 1 + \sigma_1 x^{-1} + \sigma_2 x^{-2} + \ldots + \sigma_t x^{-t} \qquad (2.30)$$

Untuk $\sigma(x) = 0$, persamaan (2.30) dapat ditulis kembali menjadi :

$$\sigma_1 x^{-1} + \sigma_2 x^{-2} + \ldots + \sigma_t x^{-t} = 1 \qquad (2.31)$$

Dengan perjanjian bahwa simbol *codeword* dengan pangkat tertinggi dikirim pertama kali, maka lokasi yang pertama kali dites adalah lokasi α^{n-1}. Pengetesan / evaluasi α^{n-1} untuk mencari akar yang sesuai, sama dengan evaluasi untuk mencari kesamaan berikut :

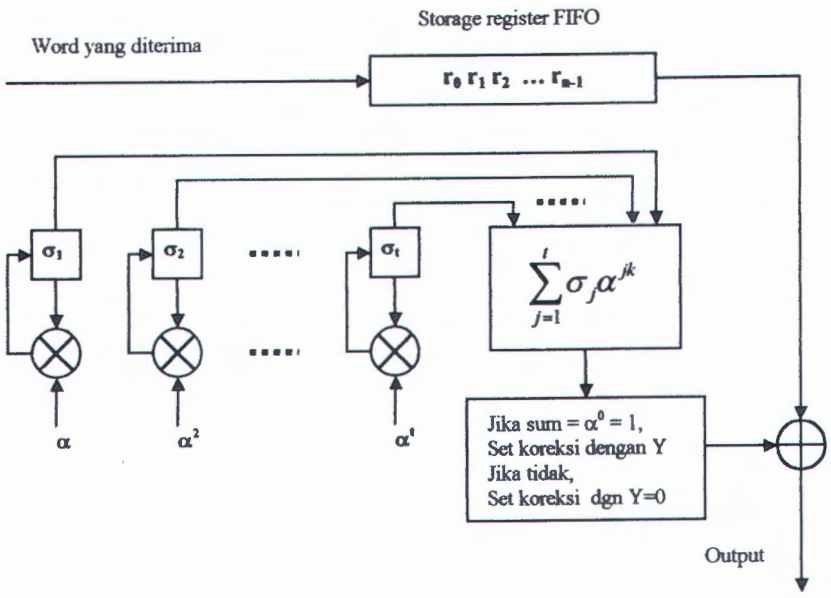
$$\sigma_1 \alpha + \sigma_2 \alpha^2 + \ldots + \sigma_t \alpha^t \stackrel{?}{=} 1 \qquad (2.32)$$

Secara umum, pengetesan α^{n-k} sebagai *error locator* ekivalen dengan mendapatkan α^k yang memenuhi persamaan :

$$\sum_{j=1}^{t} \sigma_j \alpha^{jk'} = \alpha^0 = 1 \qquad , j = 0, 1, 2, \ldots, n-1 \qquad (2.33)$$

Diagram blok *Chien search* ditunjukkan dalam gambar 2.8. *Rangkaian Chien search* melangkah secara berurutan ke seluruh nilai lokasi simbol. Jika ditemukan simbol yang salah, maka dilakukan perbaikan. Di permulaan langkahnya, register $\sigma(x)$ diinisialisasi dengan koefisien yang diperoleh dari metode Peterson. Setelah itu, rangkaian dijalankan selama n langkah, sesuai dengan panjang blok kode. Tiap operasi perkalian di tiap langkah, jumlah dari isi register $\sigma(x)$ dibandingkan dengan value 1. Jika nilainya sama dengan 1, maka simbol yang bersangkutan mengandung *error*, dan harus diperbaiki.

Jika *word* yang diterima mengandung kesalahan sebanyak t atau lebih kecil, di mana t adalah batas maksimal kode yang dapat dikoreksi, maka *Peterson's direct solution* akan menghasilkan koefisien untuk polinomial *error locator*. Selanjutnya *Chiens search* akan mencari akar polinomial, dan memperbaiki *error* di tempat yang



Gambar 2.8. Diagram blok Chien search

sesuai. Tetapi, jika *Chien search* menghasilkan jumlah akar yang lebih kecil dari seharusnya, maka polinomial tidak mempunyai akar yang sesuai dengan *field locator*. Dengan demikian, polinomial tersebut bukan merupakan polinomial *error locator* yang benar.

Kasus semacam itu, sebagian besar mengindikasikan bahwa telah terjadi deteksi yang tidak tepat terhadap pola *error* yang muncul. Pola tersebut mengandung lebih dari t *error*, tetapi tidak mengakibatkan proses *decoding* ke *codeword* yang salah. Dalam beberapa kasus, pola yang mengandung lebih dari t *error* luput dari deteksi *Chien search*, sehingga diperlukan metode pemeriksaan yang lebih baik. Ketika *Peterson's direct solution* mengindikasikan munculnya *error* dengan $l < t$ dan *Chien search* melakukan koreksi l digit, *codeword* yang dihasilkan harus diperiksa untuk memastikan bahwa persamaan sindromnya memenuhi. Ketidakcocokan pada persamaan sindrom menunjukkan adanya deteksi yang tidak tepat terhadap pola

error yang bersangkutan. Untuk koreksi t *error*, seluruh persamaan sindrom akan digunakan oleh *decoder* dan sindrom *codeword* hasil *decoding* tidak perlu diperiksa.

2.6. FPGA dan Xilink Foundation F2.1i

FPGA (Field Programmable Gate Array) merupakan sebuah chip yang dapat diprogram untuk menjadi rangkaian yang diinginkan. FPGA diproduksi oleh beberapa produsen. Salah satu produsen FPGA terlaris adalah Xilink Corporation. Ada beberapa jenis FPGA yang diproduksi, salah satunya adalah tipe XC4010XLPC84 yang digunakan dalam tesis ini. Agar dapat digunakan, Chip FPGA dilengkapi dengan komponen-komponen tambahan sehingga membentuk sebuah *prototipe board.*

Untuk memprogram *prototipe board* , diperlukan perangkat lunak. Salah satunya adalah Xilink Foundation F2.1i. Perangkat lunak ini sangat membantu proses rancang bangun rangkaian digital yang diimplementasikan di FPGA. Berikut ini, penjelasan lebih rinci tentang chip FPGA dan perangkat lunak Xilink Foundation F2.1i.

2.6.1. Field Program Gate Array

Field Programmable Gate Array adalah rangkaian terintegrasi yang mengandung puluhan hingga puluhan ribu sel logik yang identik dengan gerbang / gate standar. Variasi tiap gerbang cukup banyak. Untuk gerbang-gerbang dasar, variasi dilakukan pada jumlah dan jenis masukan. Jumlah masukan biasanya mulai dari 2 hingga 10 buah. Jenis masukannya bisa berupa logika aktif positif atau negatif.

Sebelum diprogram, tiap sel tidak terikat satu dengan lain. Keterikatan hanya terjadi jika program dimasukkan untuk membuat koneksi antara satu sel dengan sel lainnya. Koneksi dilakukan dengan menggunakan sambungan-sambungan atau saklar-saklar dalam FPGA yang umumnya disusun dalam bentuk matriks.

FPGA tidak hanya mengakomodasi kebutuhan akan gerbang dasar saja, tetapi juga rangkaian logika kombinasi seperti *decoder, multiplexer, adder*. Untuk rangkaian logika kombinasi, secara fisik rangkaian tidak diimplementasikan dalam bentuk rangkaian gerbang, melainkan *look up table* (LUT). Selain rangkaian logika kombinasi, FPGA juga dapat mengakomodasi kebutuhan akan rangkaian logika sekuensial mulai dari flip-flop (JK, RS, T, D) hingga *counter, shift register.*

Pemakaian FPGA mempunyai banyak keuntungan dibanding pemakaian rangkaian terintegrasi jenis SSI, MSI jenis TTL (74 .. , 54 ..) , CMOS (4..). Selain jumlah gerbangnya yang jauh lebih banyak, kemudahan, kecepatan dan fleksibilitas proses rancang bangun merupakan kelebihan FPGA. Untuk menampilkan kelebihannya, perangkat keras FPGA harus ditunjang oleh perangkat lunak. Untuk chip FPGA Xilink XL4010PC84, perangkat lunak yang dapat digunakan adalah Xilink Foundation F2.1i.

2.6.2. Xilink Foundation F2.1i

Jika suatu rancangan skema rangkaian logika telah didapat, skema tersebut harus diterjemahkan dalam bahasa VHDL (Visual Hardware Description Language). Program dalam Bahasa VHDL merupakan sekelompok instruksi yang menggambarkan fungsi/gerbang logika dasar dan fungsi-fungsi lainnya. Pada Xilink Foundation F2.1i bahasa VHDL dapat ditulis pada fasilitas bernama HDL Editor.

Bagi sebagian kalangan, membuat rangkaian dengan bahasa VHDL merupakan pekerjaan yang merepotkan. Karena itu, software Xilink Foundation F2.1i menyediakan fasilitas Schematic Editor yang digunakan untuk menggambar rangkaian dalam bentuk skematis, dengan menggunakan simbol-simbol fungs-fungsi logika yang familiar. Fasilitas editor lainnya adalah FSM Editor yang digunakan untuk menggambar rangkaian dengan menggunakan simbol-simbol Finite State Machine.

Rangkaian yang telah dibuat dalam salah satu dari ketiga bentuk di atas, dapat disimulasi dengan menggunakan fasilitas Logic Simulator. Hasil simulasi ditampilkan dalam bentuk diagram waktu dari masukan-masukan, keluaran-keluaran, atau titik-titik pengamatan yang dipilih.

Jika hasil simulasi telah sesuai dengan yang diinginkan, file program rangkaian diterjemahkan ke bentuk file yang dapat dimengerti chip FPGA. Penterjemahan dengan menggunakan fasilitas Implementasi. Program ini akan melakukan proses : *translate, map, place & route, timing simulation*, dan *configure*. Dan pada akhirnya akan dihasilkan file berekstensi BIT, yang dapat di-loading ke FPGA dengan menggunakan fasilitas komunikasi PC seperti: format paralel LPT 1:, format serial RS 232, atau USB. Jika file program telah selesai di-loading, maka hubungan komunikasi dapat dicabut dan FPGA aktif secara mandiri.

BAB III

RANCANG BANGUN CODEC REED-SOLOMON PADA FPGA

3.1. Penentuan Parameter Kode RS

Untuk mendapatkan parameter n dan k pada kode RS (n,k), parameter m dan t harus dipilih dahulu. Selain menghasilkan parameter n dan k, pemilihan parameter m dan t juga menghasilkan parameter lain, seperti: jarak minimum d_{min} dan polinomial generator g(x).

Penentuan parameter kode RS dilakukan dengan menggunakan langkah-langkah berikut:

a) Mengamati m dan t kode RS yang sering digunakan. Jumlah bit persimbol yang biasa digunakan pada pengiriman data berkecepatan tinggi adalah m = 8 bit = 1 byte, yang sesuai dengan jumlah bit data komputer digital. Dengan m = 8, akan diperoleh panjang kode $n = 2^m+1 = 255$. Selain itu juga diperlukan 8 set *encoder* bit, 8 kanal (pengiriman paralel) atau 8 clock untuk pengiriman tiap simbol secara serial. Panjang kode lain yang dapat digunakan adalah n = 207 (pada HDTV), n = 236, n = 32, juga n = 15. Untuk n = 15, diperoleh m = 4 bit = 1 nibble. Untuk m=4, set *encoder*, kanal dan *decoder* yang diperlukan menjadi ½ kali dari set dengan m = 8 bit. Untuk batasan kemampuan koreksi, nilai t yang digunakan cukup beragam. Nilai yang sering digunakan adalah t = 8 . HDTV menggunakan t = 10. Nilai lainnya adalah t = 2, t = 3. Nilai t yang besar akan

memperpanjang: deretan memori yang dipasang di *encoder*, deretan penghitung sindrom (*syndrome calculator*), dan bagian lain di *decoder*.

b) Mengamati spesifikasi FPGA yang digunakan. FPGA Xilink XC4010XLPC84 mempunyai data sebagai berikut :

- Level tegangan :

  masukan : 0 V (logika 0), 3,3 – 5,5 V (logika 1)

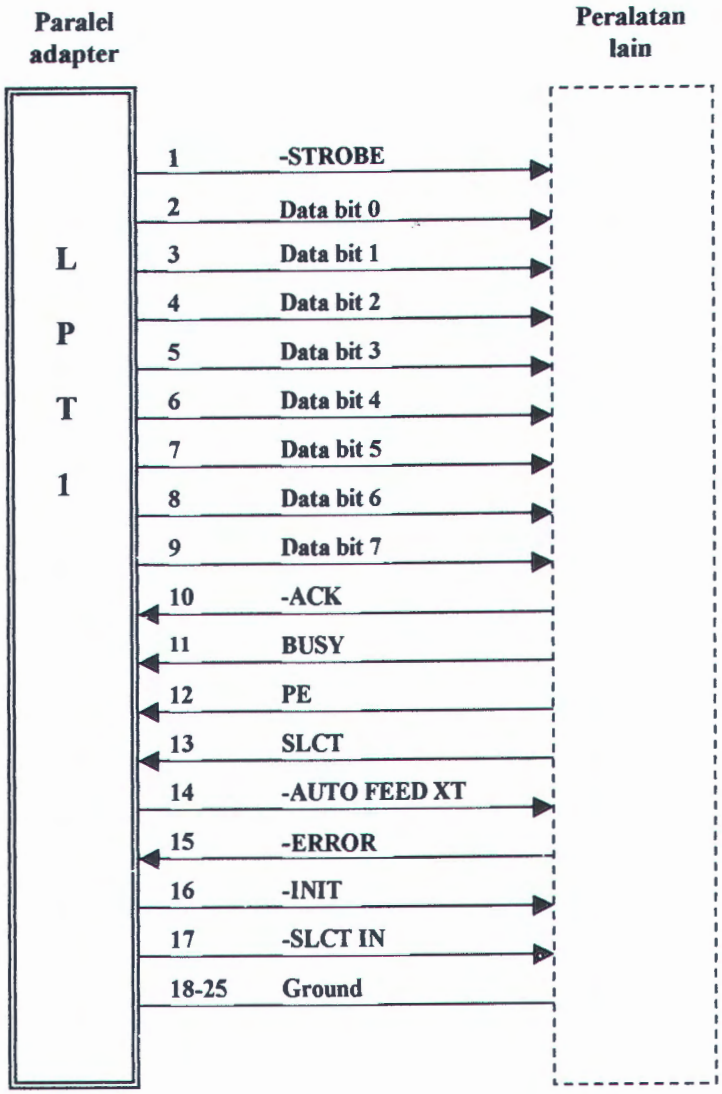
  keluaran : 0 V (logika 0), 3,3 – 3,4 V (logika 1)

- Tegangan Suplai 3,3 Volt

c) Mengamati spesifikasi peralatan komunikasi radio dan modem yang digunakan. Peralatan komunikasi radio yang digunakan adalah pemancar dan penerima FM stereo dengan TCM 3051 sebagai modem yang data-datanya adalah sebagai berikut :

- frekuensi yang digunakan adalah 100 MHz.
- modulasi yang digunakan adalah FM dengan bandwidth 200 KHz .
- laju modem maksimum (kirim dan terima) = 1200 bps
- modem menggunakan modulasi AFSK, dengan data masukan dan keluaran berupa data biner. Maka modemnya merupakan modem BFSK (Binary FSK).

d) Mengamati spesifikasi *port* komputer PC yang digunakan. *Port* LPT1: pada komputer PC, umumnya digunakan untuk berkomunikasi dengan *printer*. Konfigurasi *port*nya ditunjukkan dalam gambar 3.1.



Gambar 3.1 Konfigurasi *port* paralel LPT1:

Walaupun mempunyai jalur keluaran cukup banyak, *port* paralel LPT1: hanya mempunyai 5 jalur masukan, yaitu –ACK, BUSY, PE, SLCT, dan –ERROR.

e) Menentukan parameter kode RS berdasar tingkat kerumitan dan kemampuan peralatan. Karena *port* LPT1: hanya mempunyai 5 jalur masukan, maka jumlah bit persimbol kode RS dipilih m = 4. Satu jalur sisanya digunakan untuk sinkronisasi. Sementara itu, untuk memperkecil tingkat kerumitan, nilai t dipilih = 2, sehingga :

- Panjang *codeword* $n = 2^m - 1 = 2^4 - 1 = 15$ simbol
- Elemen-elemen kode diambil dari $GF(2^4)$
- Polinomial primitif yang digunakan adalah $p(x) = 1 + x + x^4$

Dengan panjang *codeword* 15 dan t = 2 simbol, diperoleh :

- Panjang paritas $n-k = 2t = 2 . 2 = 4$ simbol
- Panjang data masukan $k = n-2t = 15 - 4 = 11$ simbol, sehingga dengan format penulisan RS(n,k,t), kode RS yang akan dirancang bangun adalah **RS(15,11,2)**
- $d_{min} = 2t + 1 = 2.2 + 1 = 5$ simbol
- $g(x) = (x + \alpha)(x + \alpha^2) \ldots (x + \alpha^{2t}) = g_0 + g_1 x + \ldots + g_{2t} x^{2t}$

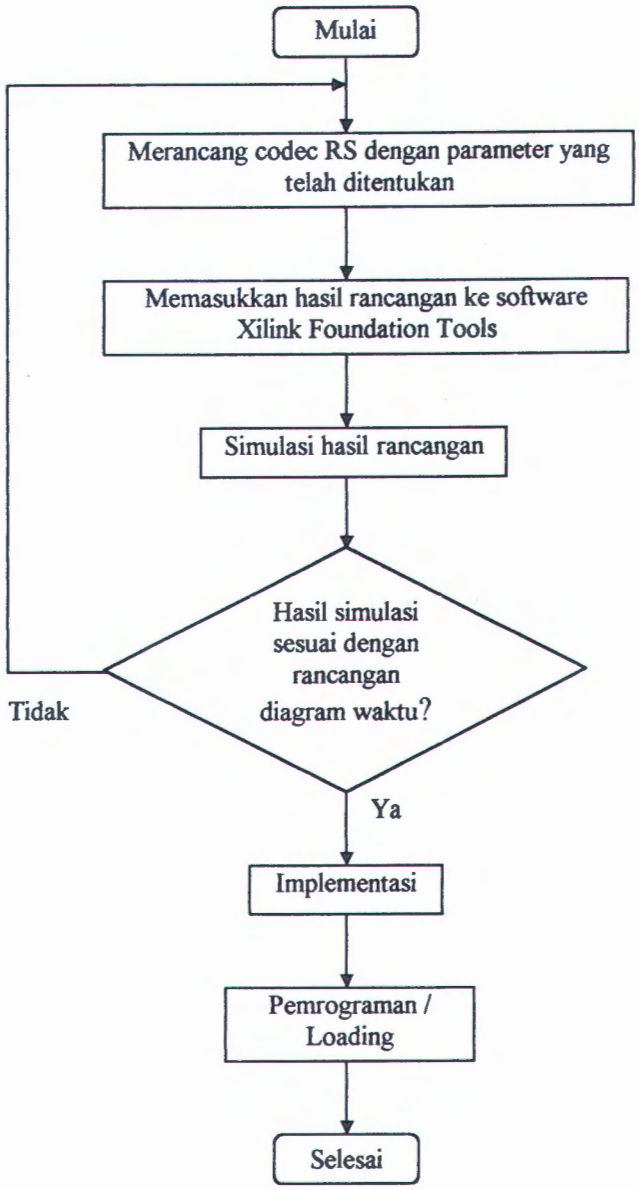
  $= (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)$

  $= g_0 + g_1 x + g_2 x^2 + g_3 x^3 + g_4 x^4$

  Penghitungan lebih lanjut dilakukan di sub bab 3.3.
- Laju kode = k/n = 11 / 15

3.2. Diagram Alir Rancang Bangun Codec RS Pada FPGA

Rancang bangun *codec* RS dilakukan sesuai diagram alir (*flowchart*) dalam gambar 3.2.



Gambar 3.2. Diagram alir proses rancang bangun *codec* RS

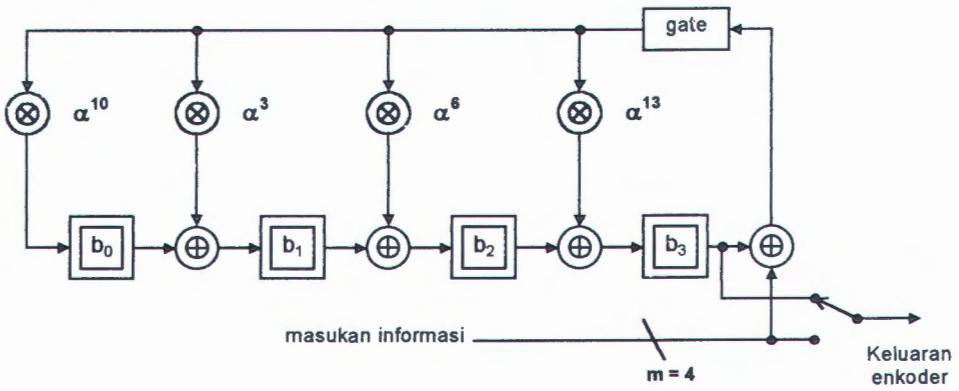
Perancangan *codec* RS dilakukan dalam dua bagian, yaitu perancangan *encoder* dan perancangan *decoder*. Perancangan *decoder* relatif lebih rumit dibanding perancangan *encoder*, karena *decoder* mempunyai blok rangkaian yang lebih banyak dan lebih kompleks.

3.3. Rancang Bangun Encoder RS

Dari sub bab 3.1. diperoleh $g(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + g_4 x^4$. Dengan perhitungan lebih lanjut menggunakan penjumlahan dan perkalian dalam $GF(2^4)$ diperoleh koefisien generator berikut.

$$g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + g_4 x^4 = \alpha^{10} + \alpha^4 x + \alpha^2 x^2 + \alpha^{13} x^3 + x^4$$

Diagram rangkaian *encoder* untuk kode RS(15,11,2) ditunjukkan dalam gambar 3.3.



Gambar 3.3 Diagram rangkaian *encoder* kode RS(15,11,2)

Tiap komponen dalam gambar 3.3 memproses 4 bit biner / 1 simbol secara paralel dengan operasi perkalian koefisien generator dalam $GF(2^4)$.

Encoder menerima 11 buah simbol, dengan memproses 4 bit simbol secara paralel. Dua hal yang dikerjakan bersama-sama hingga langkah / step ke 11, adalah pengiriman simbol ke keluaran *encoder* dan perkalian dengan generator. Karena itu, saklar pada keluaran *encoder* dihubungkan langsung dengan terminal masukan informasi. Setelah langkah ke 11, saklar keluaran *encoder* dihubungkan ke keluaran

generator. Kemudian, pada langkah 12 hingga 15, simbol paritas yang diperoleh pada langkah ke 11, dikirim keluar *encoder*.

Setelah satu periode pengkodean/1 blok, akan terkirim 15 simbol, yang tiap simbolnya mengirim 4 bit secara serentak. Untuk itu, jenis modulator yang diperlukan adalah 16-*ary*. Tetapi, karena modulator yang digunakan pada modem (untuk evaluasi *codec*) adalah *binary* AFSK, maka tiap bit dari tiap simbol, dikirim secara serial. Periode pengiriman serial adalah sebesar ¼ kali periode satu simbol. Proses multipleks dengan periode ¼ kali periode simbol ini, menyebabkan kecepatan kirim tiap simbol tidak mengalami perubahan.

3.3.1. Koefisien Pengali Generator

Koefisien pengali generator dapat diimplementasikan dalam bentuk penjumlahan biner dengan rangkaian berdasar perhitungan berikut ini. Untuk nilai m = 4, diperoleh polinomial $\beta = b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3$. Jika polinomial β dikalikan dengan α^6 dan hasilnya berupa polinomial $\gamma = c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3$, maka hubungan antara kedua polinomial ditentukan oleh persamaan :

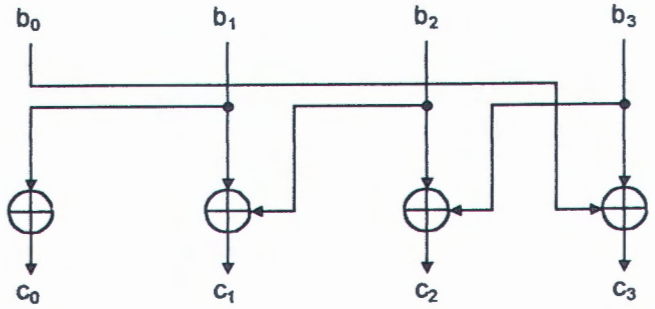
$$\gamma = \alpha^3 \beta = \alpha^3 (b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3)$$

$$= b_0 \alpha^3 + b_1 \alpha^4 + b_2 \alpha^5 + b_3 \alpha^6$$

$$= b_0 \alpha^3 + b_1 (1 + \alpha) + b_2 (\alpha + \alpha^2) + b_3 (\alpha^2 + \alpha^3)$$

$$c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 = b_1 + \alpha (b_1 + b_2) + \alpha^2 (b_2 + b_3) + \alpha^3 (b_0 + b_3)$$

Sesuai persamaan tersebut, rangkaian pengali α^3 dari GF(2^4) dapat dibuat seperti dalam gambar 3.4.

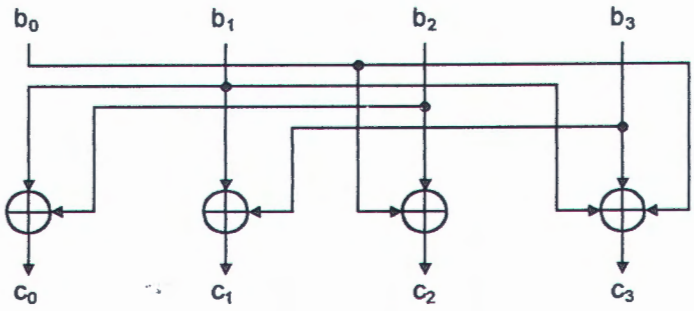


Gambar 3.4 Rangkaian pengali α^3 dari GF(2^4)

Dengan cara yang sama, persamaan dan gambar rangkaian pengali α^6 adalah :

$$\gamma = \alpha^6 \beta = \alpha^6 (b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3)$$

$$c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 = (b_1 + b_2) + \alpha(b_1 + b_3) + \alpha^2 (b_0 + b_2) + \alpha^3 (b_0 + b_1 + b_3)$$



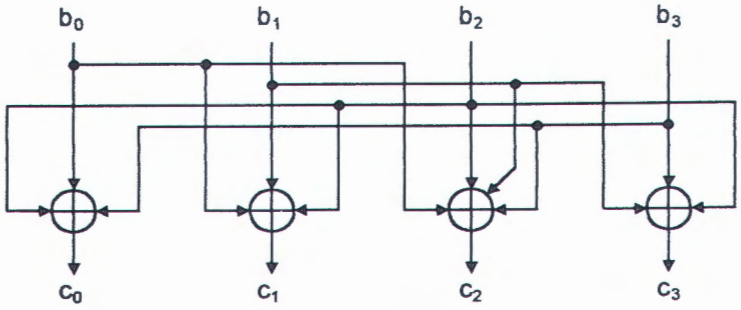
Gambar 3.5 Rangkaian pengali α^6 dari GF(2^4)

Persamaan dan gambar rangkaian pengali α^{10} adalah :

$$\gamma = \alpha^{10} \beta = \alpha^{10} (b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3)$$

$$c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 = (b_0 + b_2 + b_3) + \alpha (b_0 + b_1 + b_2) + \alpha^2 (b_0 + b_1 + b_2 + b_3)$$

$$+ \alpha^3 (b_1 + b_2 + {}_3)$$

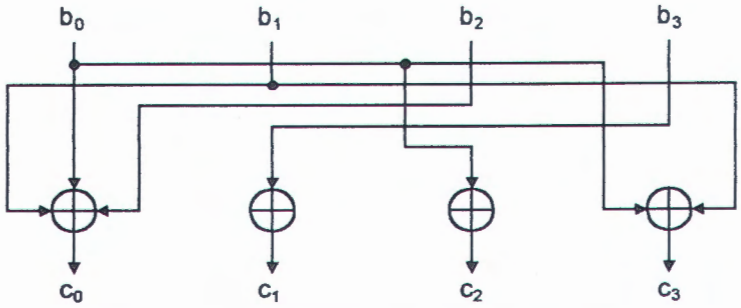


Gambar 3.6 Rangkaian pengali α^{10} dari GF(2^4)

Persamaan dan gambar rangkaian pengali α^{13} adalah :

$$\gamma = \alpha^{13}\,\beta = \alpha^{13}(b_0 + b_1\,\alpha + b_2\,\alpha^2 + b_3\,\alpha^3)$$

$$c_0 + c_1\,\alpha + c_2\,\alpha^2 + c_3\,\alpha^3 = (b_0 + b_1 + b_2) + \alpha\,b_3 + \alpha^2\,b_0 + \alpha^3\,(b_0 + b_1)$$

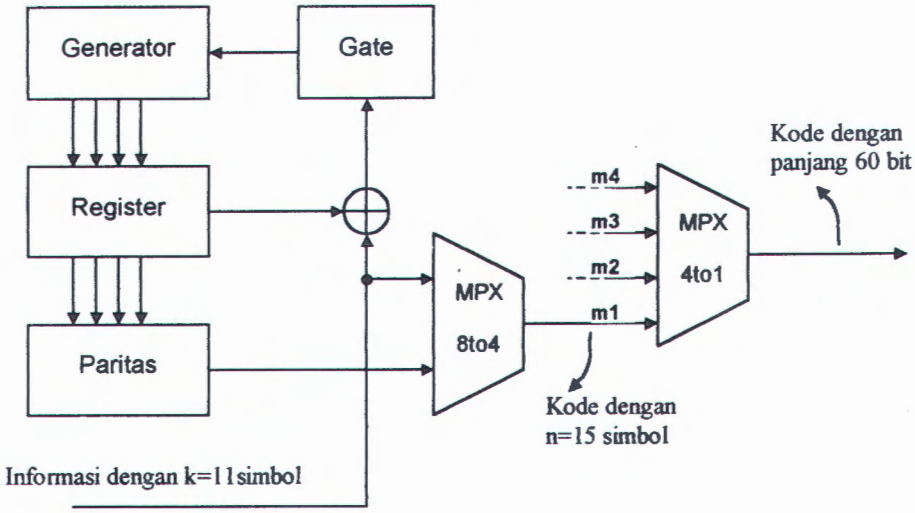


Gambar 3.7 Rangkaian pengali α^{13} dari GF(2^4)

3.3.2. Perancangan Diagram Waktu

Untuk memperjelas urutan kerja proses *encoding*, pada gambar 3.8 ditunjukkan diagram blok *encoder*. Rancangan diagram waktu proses *encoding* ditunjukkan dalam gambar 3.9. Tiap generator, *gate*, register, paritas, penjumlah dan multiplekser MPX8to4 terdiri dari empat unit yang bekerja secara simultan sehingga diperoleh 4 bit keluaran yang paralel. Untuk mendapatkan bentuk serialnya (dengan

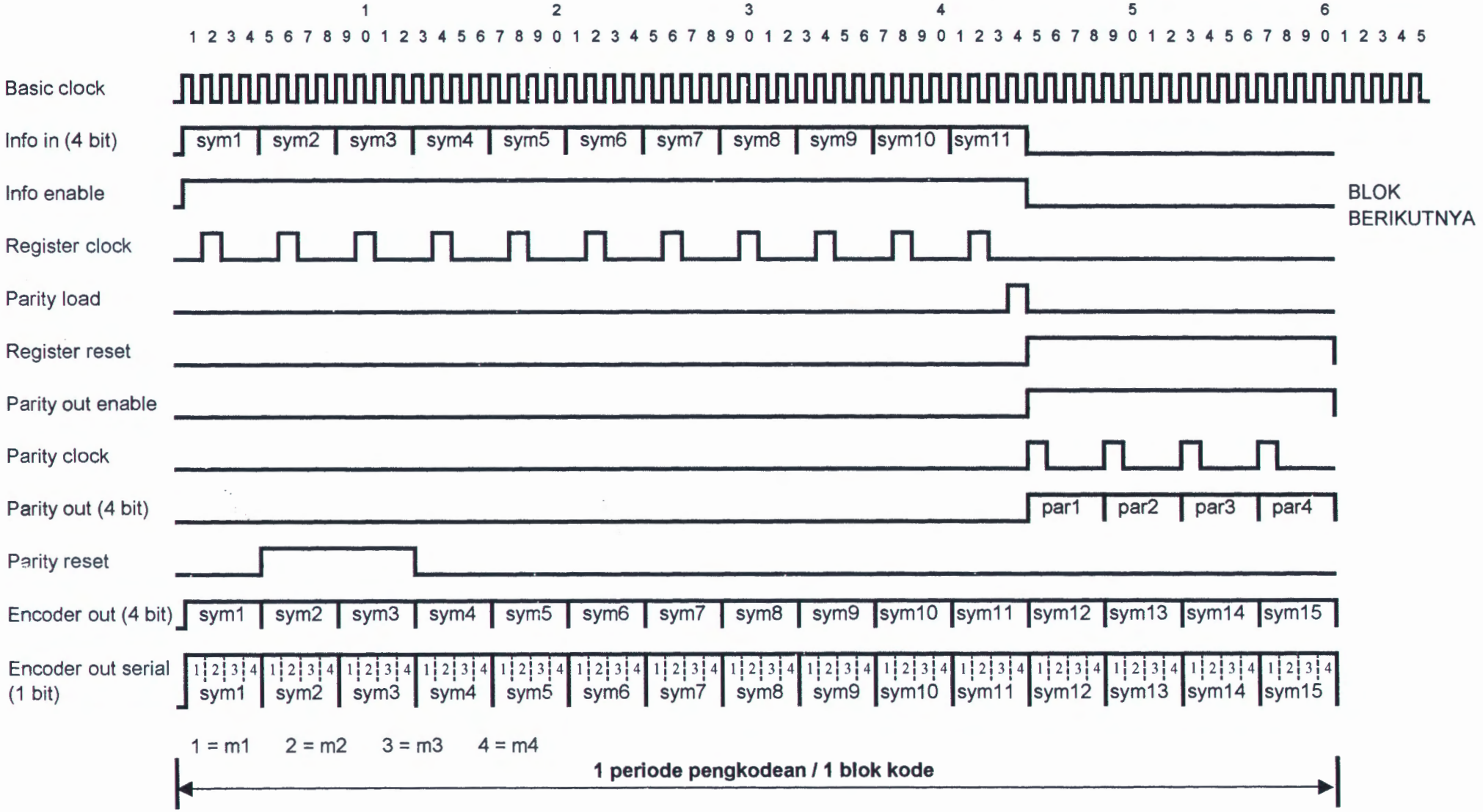
panjang 60 bit), digunakan multiplekser MPX4to1 yang mempunyai kecepatan 4 kali kecepatan informasi yang masuk.



Gambar 3.8 Diagram blok *encoder*

Dalam gambar 3.9 tampak bahwa tiap simbol diterima dan dikirim dalam 4 satuan waktu dasar (*basic clock*). Hal itu dilakukan untuk mengakomodasi proses multipleks simbol 4 bit paralel menjadi serial. Nilai periode *basic clock* ditentukan berdasar kecepatan kirim modem yang mempunyai nilai maksimum 1200 bps. Nilainya diperoleh dari perhitungan berikut ini.

Sinyal yang akan dikirim melalui modem adalah *basic clock* (yang berfungsi sebagai sinkronisasi) dan sinyal serial dari data paralel 4 bit. Satu periode *basic clock*, terdiri dari setengah periode logika positif dan setengah periode logika nol. Ini berarti satu periode *clock* mengandung 2 bit. Dengan kecepatan modem maksimum sebesar 1200 bps, maka frekuensi *basic clock* maksimum adalah 1200/2 = 600 Hz dan periodenya adalah 1 / 600 = 1,67 ms. Agar nilainya bulat dan kecepatan data di bawah maksimum, digunakan periode *basic clock* sebesar 2 ms.



Gambar 3.9 Rancangan diagram waktu proses encoding

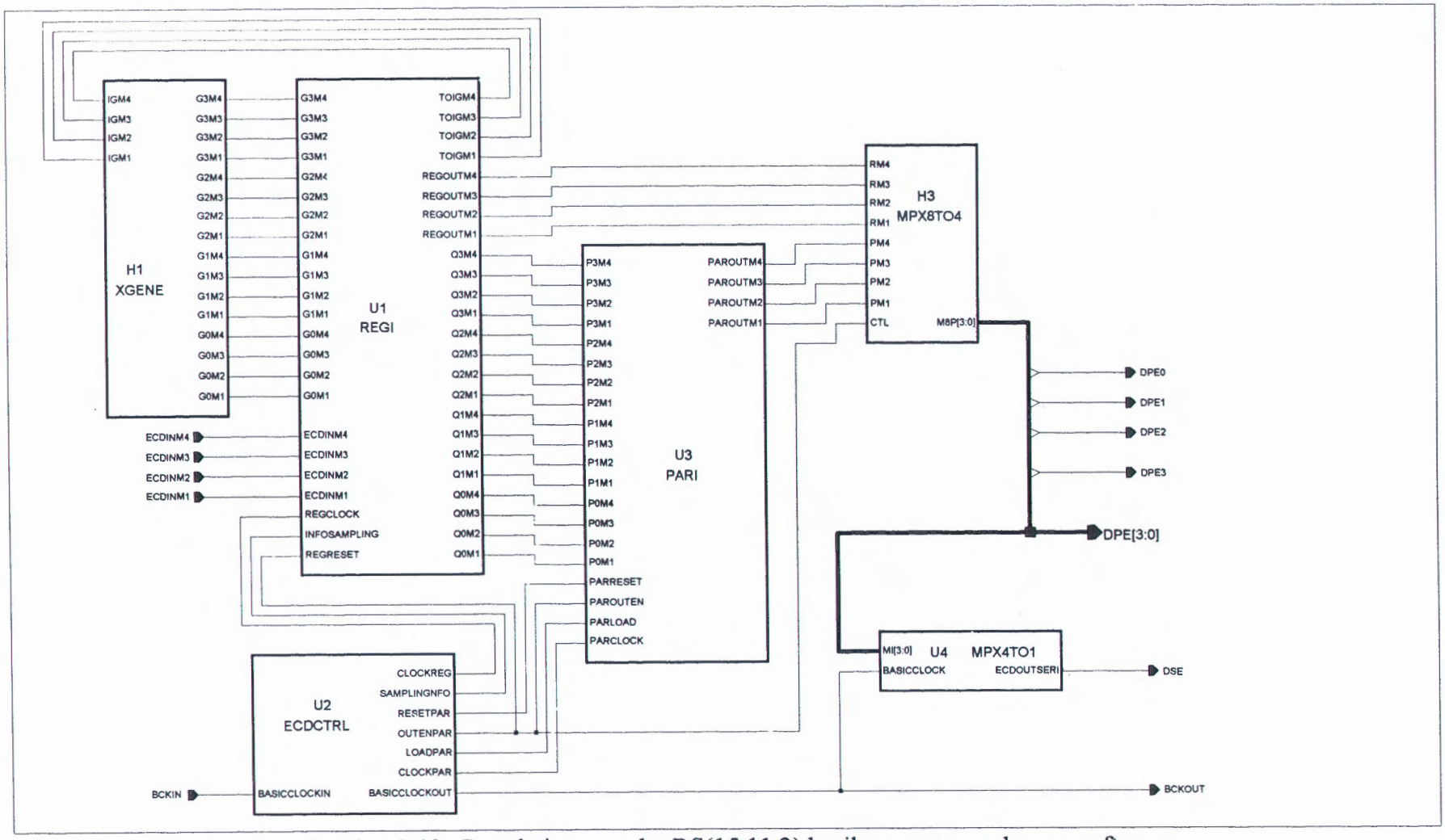
Informasi 1 simbol (4 bit) dengan panjang k=11 masuk ke *encoder* secara berurutan dari satu simbol ke simbol berikutnya. Multiplekser MPX8to4 langsung mengeluarkan informasi yang masuk, sedangkan *gate* membuka dirinya untuk mengambil tiap simbol yang masuk. Simbol yang diambil, langsung dimasukkan ke unit generator untuk dikalikan dengan koefisien g_0 hingga g_3. Hasil perkalian merupakan paritas yang disebabkan oleh masuknya tiap satu simbol.

Tiap menerima satu simbol, register *encoder* akan melakukan satu kali penggeseran disertai dengan operasi penjumlahan biner. Pada akhir simbol informasi ke 11, isi register *encoder* (sebesar 4 simbol) yang merupakan paritas dari informasi masuk, dipindahkan ke blok paritas untuk disimpan secara simultan. Setelah itu, simbol paritas dikeluarkan satu persatu secara berurutan.

Selanjutnya, multiplekser 4 ke 1 kanal akan mengeluarkan bit-bit informasi dan paritas dengan urutan : m1, m2, m3, dan m4. Satu putaran pengeluaran data m1 hingga m4 dilakukan tiap diterima 1 simbol.

3.3.3. Membuat Rangkaian Encoder di Software Xilink Foundation F2.1i

Rangkaian *encoder* disusun dengan mempertimbangkan karakteristik diagram waktu dari komponen yang dipakai dalam rangkaian (flip-flop D, flip-flop T, gerbang XOR, dan gerbang AND). Penyusunan dilakukan dengan *software* Xilink Foundation F2.1i. Hasil penyusunan ditunjukkan dalam gambar 3.10. Rangkaian dalam gambar 3.10 merupakan rangkaian dari simbol-simbol makro yang dibentuk oleh gerbang-gerbang logika dasar, flip-flop T , dan flip-flop D.



Gambar 3.10 Rangkaian encoder RS(15,11,2) hasil penyusunan dengan software

3.3.4. Simulasi Encoder RS(15,11,2) dan Pemeriksaan Hasilnya

Proses simulasi dilakukan dengan *software* yang sama. Diagram waktu hasil simulasi ditunjukkan dalam gambar 3.11. Untuk memeriksa kebenaran rangkaian, dimasukkan sebuah informasi dengan persamaan polinomial :

$$u(x) = x^{10} + x^9 + \alpha x^7 + \alpha^2 x^6 + \alpha^3 x^5 + \alpha^4 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^8 x + \alpha^9$$

*Codeword*nya diperoleh dengan langkah perhitungan sebagai berikut :

Step 1. Mengalikan informasi u(x) dengan $x^{15-11} = x^4$, dan diperoleh :

$$u(x)\,x^4 = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

Step 2. Mendapatkan paritas p(x) dari sisa pembagian u(x) x^4 terhadap g(x). Proses perhitungannya adalah sebagai berikut :

$$\underline{x^{10} + \alpha^6 x^9 + \alpha^{12} x^8 + \alpha x^7 + \alpha^3 x^6 + \alpha x^5 + \alpha^3 x^4 + \alpha^{13} x^3 + \alpha^5 x^2 + \alpha^6 x + \alpha^3}$$

$$x^4 + \alpha^{13} x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10} \,|\, x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

$$\underline{x^{14} + \alpha^{13} x^{13} + \alpha^6 x^{12} + \alpha^3 x^{11} + \alpha^{10} x^{10}}$$

$$\alpha^6 x^{13} + \alpha^6 x^{12} + \alpha^9 x^{11} + \alpha^4 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

$$\underline{\alpha^6 x^{13} + \alpha^4 x^{12} + \alpha^{12} x^{11} + \alpha^9 x^{10} + \alpha x^9}$$

$$\alpha^{12} x^{12} + \alpha^8 x^{11} + \alpha^{14} x^{10} + \alpha^9 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

$$\underline{\alpha^{12} x^{12} + \alpha^{10} x^{11} + \alpha^3 x^{10} + x^9 + \alpha^7 x^8}$$

$$\alpha x^{11} + x^{10} + \alpha^7 x^9 + \alpha^3 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

$$\underline{\alpha x^{11} + \alpha^{14} x^{10} + \alpha^7 x^9 + \alpha^4 x^8 + \alpha^{11} x^7}$$

$$\alpha^3 x^{10} + \quad + \alpha^7 x^8 + \alpha^3 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

$$\underline{\alpha^3 x^{10} + \alpha x^9 + \alpha^9 x^8 + \alpha^6 x^7 + \alpha^{13} x^6}$$

$$\alpha x^9 + x^8 + \alpha^2 x^7 + x^6 + \alpha^8 x^5 + \alpha^9 x^4$$

$$\underline{\alpha x^9 + \alpha^{14} x^8 + \alpha^7 x^7 + \alpha^4 x^6 + \alpha^{11} x^5}$$

$$\alpha^3 x^8 + \alpha^{12} x^7 + \alpha x^6 + \alpha^7 x^5 + \alpha^9 x^4$$

$$\underline{\alpha^3 x^8 + \alpha x^7 + \alpha^9 x^6 + \alpha^6 x^5 + \alpha^{13} x^4}$$

$$\alpha^{13} x^7 + \alpha^3 x^6 + \alpha^{10} x^5 + \alpha^{10} x^4$$

$$\underline{\alpha^{13} x^7 + \alpha^{11} x^6 + \alpha^4 x^5 + \alpha x^4 + \alpha^8 x^3}$$

$$\alpha^5 x^6 + \alpha^2 x^5 + \alpha^8 x^4 + \alpha^8 x^3$$

$$\underline{\alpha^5 x^6 + \alpha^3 x^5 + \alpha^{11} x^4 + \alpha^{8\ 3} + x^2}$$

$$\alpha^6 x^5 + \alpha^7 x^4 + x^2$$

$$\underline{\alpha^6 x^5 + \alpha^4 x^4 + \alpha^{12} x^3 + \alpha^9 x^2 + \alpha x}$$

$$\alpha^3 x^4 + \alpha^{12} x^3 + \alpha^7 x^2 + \alpha x$$

$$\underline{\alpha^3 x^4 + \alpha x^3 + \alpha^9 x^2 + \alpha^6 x + \alpha^{13}}$$

$$\boldsymbol{\alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}}$$

diperoleh $p(x) = \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$

Step 3. Mendapatkan polinomial *codeword* dengan menggabungkan x^{n-k} u(x) dan p(x)$\Rightarrow$ $c(x) = x^{n-k} u(x) + p(x)$, dan hasilnya adalah :

$$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$$

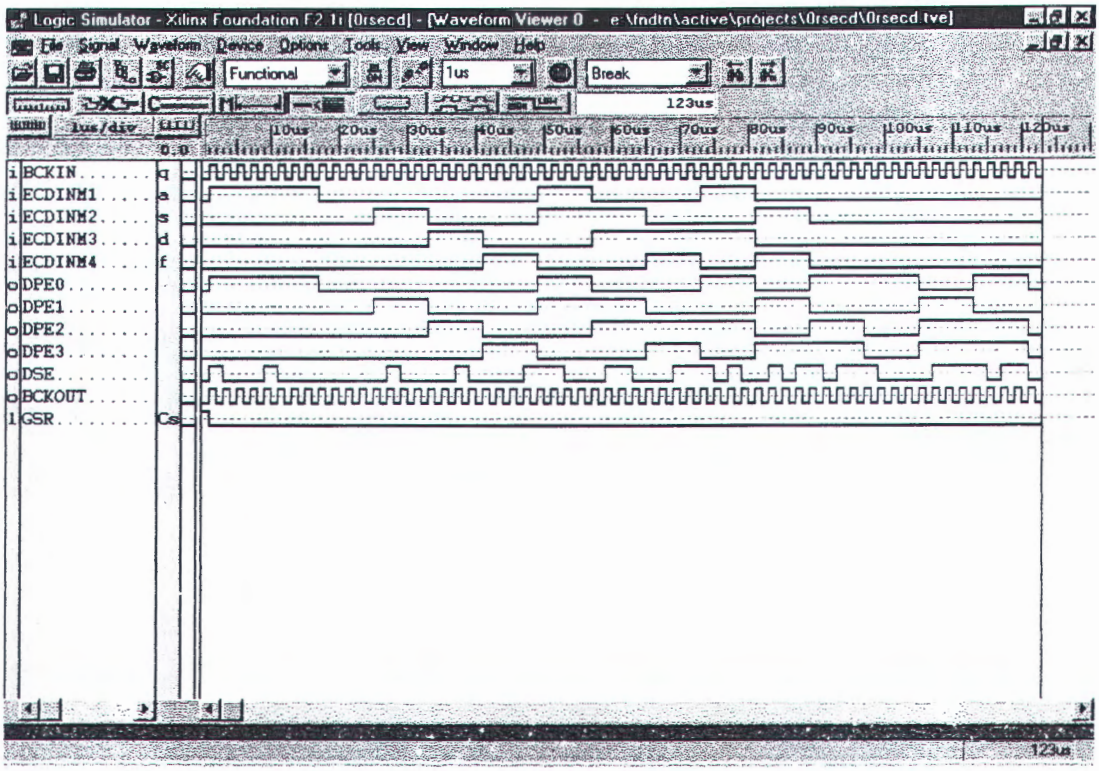
Untuk membandingkan dengan hasil simulasi, polinomial informasi dan *codeword* harus diubah dalam bentuk *tuple* biner 4 bit. Representasi *tuple* biner 4 bit untuk polinomial informasi u(x) adalah :

Info ke	1	2	3	4	5	6	7	8	9	10	11
m1	1	1	0	0	0	0	1	0	0	1	0
m2	0	0	0	1	0	0	1	1	0	0	1
m3	0	0	0	0	1	0	0	1	1	1	0
m4	0	0	0	0	0	1	0	0	1	0	1

Sedangkan representasi *tuple* biner 4 bit untuk *codeword* c(x) adalah :

Tuple ke	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
m1	1	1	0	0	0	0	1	0	0	1	0	1	1	0	1
m2	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0
m3	0	0	0	0	1	0	0	1	1	1	0	1	0	1	1
m4	0	0	0	0	0	1	0	0	1	0	1	1	0	1	1

Hasil perhitungan ini dibandingkan dengan logika keluaran DPE0, DPE1, DPE2 dan DPE3 dalam gambar 3.11. Perlu diingat bahwa tiap *tuple* pada gambar 3.11 dikeluarkan dalam waktu 4 basic clock. Perbandingan menunjukkan bahwa hasilnya adalah sama.



Gambar 3.11 Diagram waktu hasil simulasi *encoder* RS(15,11,2)

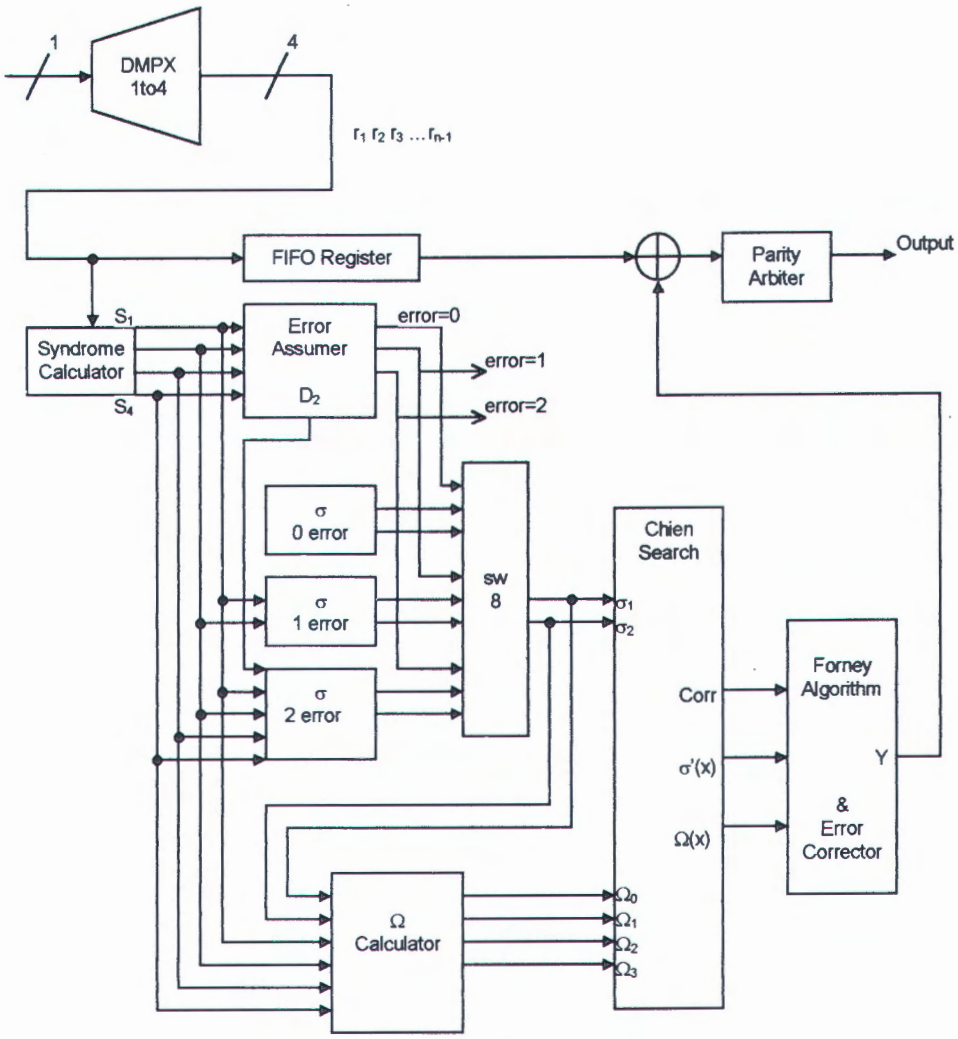
3.4. Rancang Bangun Decoder RS

Untuk kode RS(15,11,2), *decoder* yang akan dibangun, bekerja berdasarkan urutan langkah sebagai berikut :

1. menghitung sindrom (S_1, S_2, S_3, S_4)
2. menghitung koefisien *error locator polynomial* (σ_1, σ_2) dengan menggunakan metode Peterson Gorenstein Zierler
3. menghitung koefisien *error magnitude polynomial* berdasar nilai sindrom dan koefisien *error locator polynomial* yang telah diperoleh
4. mencari lokasi *error* dan nilai *error magnitude* dengan menggunakan *Chien search.*
5. menghitung *error value* dengan menggunakan algoritma Forney.

6. melakukan perbaikan dengan menjumlahkan *error value* terhadap simbol yang mengalami *error*.
7. memisahkan simbol informasi dari simbol paritasnya.

Diagram blok *decoder* RS(15,11,2) ditunjukkan dalam gambar 3.12. Pada gambar ini semua jalur mempunyai lebar 4 bit, kecuali: masukan demultiplekser DMPX1to4, keluaran *error assumer*, dan keluaran Corr (*correction*) pada *Chien search*.



Gambar 3.12 Diagram blok *decoder* RS(15,11,2)

Demultiplekser DMPX1to4 mengubah data serial yang diterima *decoder* ke bentuk paralel 4 bit. Data paralel 4 bit ini sesuai dengan jumlah bit persimbol yang dikirim oleh *encoder*.

Setelah melewati demultiplekser, data simbol-simbol yang masuk langsung diarahkan ke *syndrome calculator* dan register FIFO (*First In First Out*). Data yang masuk ke *syndrome calculator* digunakan untuk melakukan penghitungan nilai *error locator* dan *error value* tiap blok kode yang diterima. Data yang masuk ke register FIFO merupakan blok kode yang akan dikoreksi bila terjadi *error*.

Nilai sindrom yang dikeluarkan oleh *syndrome calculator* langsung diterima oleh *error assumer*, σ *calculator* (koefisien *error locator polynomial*) untuk *error* = 1 dan 2, serta Ω *calculator* (koefisien *error magnitude polynomial*) untuk *error* = 1 dan 2. σ *calculator* untuk *error* = 0 tidak memerlukan nilai sindrom, karena koefisien σ_1 dan σ_2 yang dikeluarkannya adalah 0.

Berdasar sindrom yang diterima, *error assumer* akan menilai berapa *error* yang dianggap terjadi, kemudian mengeluarkan sinyal *error* yang sesuai. Penilaian dilakukan sesuai syarat-syarat dalam *Peterson direct solution*. Sinyal *error* yang keluar dari *error assumer*, digunakan untuk mengaktifkan saklar sw8 agar koefisien σ error bersangkutan dapat masuk ke *Chien search*. Blok *error assumer* juga mengeluarkan nilai determinan D_2 yang berguna untuk menghitung koefisien σ untuk *error* = 2.

Nilai sindrom dan koefisien σ digunakan untuk menghitung nilai koefisien Ω. Kemudian secara bersama-sama, koefisien σ dan Ω dimasukkan pada *Chien search* untuk mendapatkan : lokasi error, $\sigma'(x)$ dan $\Omega(x)$. Nilai $\sigma'(x)$ dan $\Omega(x)$

digunakan oleh algoritma Forney untuk mendapatkan *error value* Y. *Chien search* akan memeriksa tiap simbol secara berurutan hingga lengkap satu blok. Jika ditemukan adanya *error* pada lokasi tertentu, ia akan mengaktifkan sinyal Corr (*Correction*) untuk mengijinkan nilai Y dikirim pada *summing point*. Jika sinyal Corr tidak aktif, nilai Y yang keluar dari *error corrector* adalah 0 (nol).

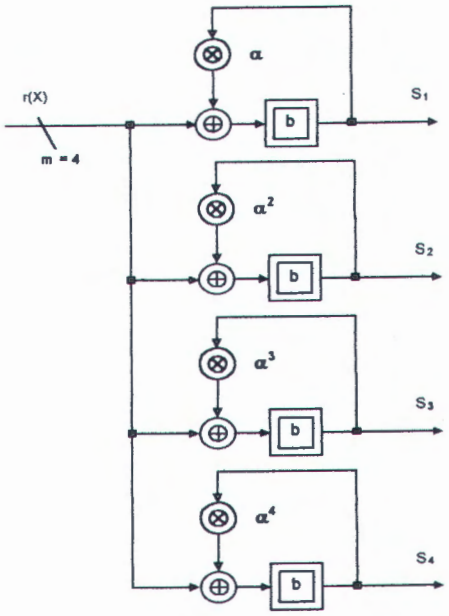
Setelah keluar dari summing point, simbol informasi dari tiap blok dipisahkan dari simbol paritasnya. Prosesnya dilakukan dengan cara mengambil informasi pada clock 1 hingga 11 dan melepas simbol paritas yang muncul pada clock 12 hingga 15. Dengan cara ini, simbol informasi dapat diperoleh kembali.

3.4.1. Syndrome Calculator

Syndrome calculator pada rangkaian decoder RS(15,11,2) akan mengeluarkan nilai sindrom untuk tiap blok kode, jika ia telah menerima 15 simbol secara lengkap. Untuk t = 2, sindrom dihitung hingga komponen ke 2t = 2.2 = 4. Jadi komponen sindrom yang dihitung adalah S_1, S_2, S_3, dan S_4, yang masing-masing adalah :

$$S_1 = r(\alpha),\ S_2 = r(\alpha^2),\ S_3 = r(\alpha^3),\ S_4 = r(\alpha^4)$$

Dengan persamaan di atas, maka diagram blok *syndrome calculator*nya ditunjukkan dalam gambar 3.13.

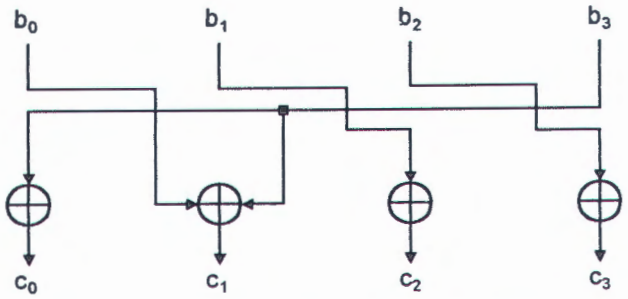


Gambar 3.13 Diagram blok *syndrome calculator*

Pembentukan rangkaian pengali GF(2^4) untuk *syndrome calculator* serupa dengan pengali pada koefisien generator *encoder*. Persamaan dan rangkaian pengali α^3 sama dengan pengali di *encoder*, sedang untuk : α , α^2 , dan α^4, persamaan dan rangkaiannya dibuat dengan cara seperti berikut. Persamaan dan gambar rangkaian pengali α adalah :

$$\gamma = \alpha\,\beta = \alpha\,(b_0 + b_1\,\alpha + b_2\,\alpha^2 + b_3\,\alpha^3)$$

$$c_0 + c_1\,\alpha + c_2\,\alpha^2 + c_3\,\alpha^3 \;=\; b_3 + \alpha\,(b_0 + b_3) + \alpha^2\,b_1 + \alpha^3\,b_2$$

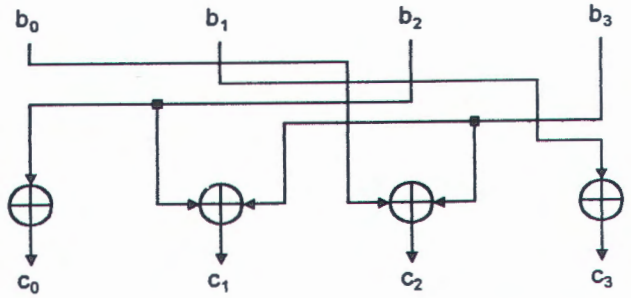


Gambar 3.14 Rangkaian pengali α dari GF(2^4)

Persamaan dan gambar rangkaian pengali α^2 adalah :

$$\gamma = \alpha^2 \beta = \alpha^2 (b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3)$$

$$c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 = b_2 + \alpha (b_2 + b_3) + \alpha^2 (b_0 + b_3) + \alpha^3 b_1$$

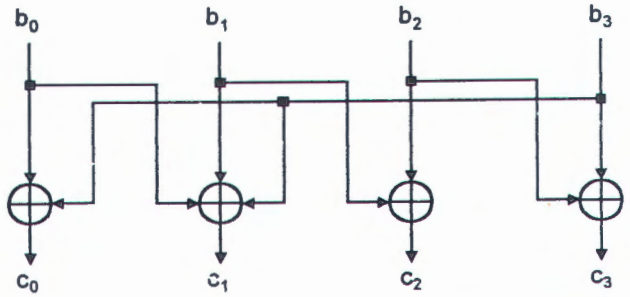


Gambar 3.15 Rangkaian pengali α^2 dari GF(2^4)

Persamaan dan gambar rangkaian pengali α^4 adalah :

$$\gamma = \alpha^4 \beta = \alpha^4 (b_0 + b_1 \alpha + b_2 \alpha^2 + b_3 \alpha^3)$$

$$c_0 + c_1 \alpha + c_2 \alpha^2 + c_3 \alpha^3 = (b_0 + b_3) + \alpha(b_0 + b_1 + b_3) + \alpha^2(b_1 + b_2) + \alpha^3(b_2 + b_3)$$



Gambar 3.16 Rangkaian pengali α^4 dari GF(2^4)

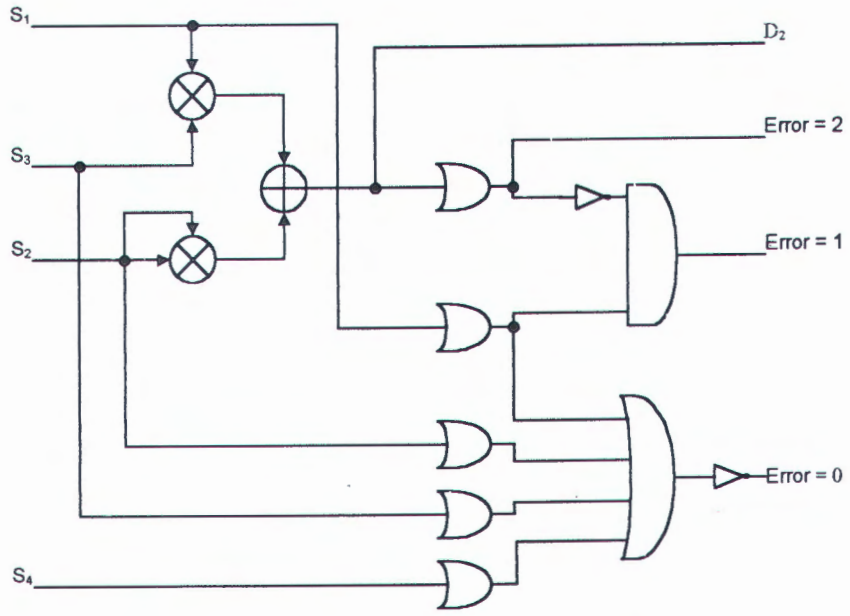
3.4.2. Error Assumer

Error assumer merupakan blok yang berfungsi untuk mendefinisikan jumlah *error* yang terjadi pada kode yang diterima. Sesuai dengan langkah kedua metode

Peterson , tiga kondisi yang mungkin terjadi untuk kemampuan koreksi *error* t sebesar 2 adalah :

1. Jika $S_i = 0$, $1 \leq i \leq 4$, maka *word* yang diterima adalah *codeword*, sehingga tidak perlu dilakukan proses koreksi
2. Jika $D_2 = S_1 S_3 + S_2^2 \neq 0$, maka dianggap terjadi dua *error*
3. Jika $D_2 = 0$ dan $S_1 \neq 0$, maka dianggap terjadi satu *error*.

Untuk t = 2 juga diperoleh persamaan determinan $D_2 = S_1 S_3 - S_2^2$. Sesuai tiga kondisi dan persamaan determinan tersebut, diagram rangkaian *error assumer*nya seperti dalam gambar 3.17. Semua jalur pada gambar 3.17 adalah 4 bit, kecuali jalur-jalur keluaran gerbang logika (*logic gate*) yang hanya selebar 1 bit.



Gambar 3.17 Diagram rangkaian error assumer

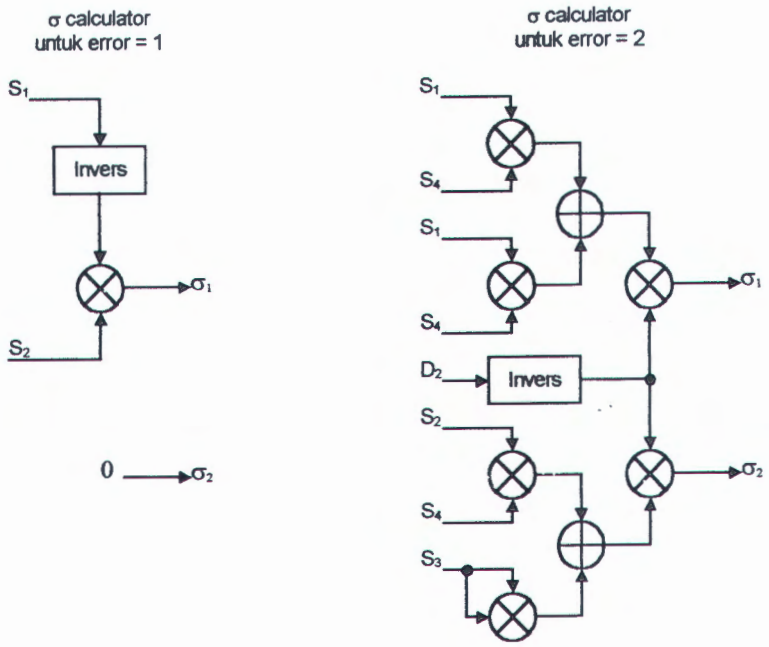
3.4.3. σ Calculator

Blok σ *calculator* berfungsi untuk menghitung koefisien polinomial *error locator* σ(x). Dengan t = 2, koefisien *error locator* yang dihitung adalah σ_1 dan σ_2.

Selain itu, untuk tiga kemungkinan *error* yang terjadi, ada tiga kemungkinan nilai koefisien yang harus dicari, yaitu :

- Untuk *error* = 0, nilai σ_1 maupun σ_2 adalah 0.
- Untuk *error* = 1, nilai $\sigma_1 = X_1 = S_2 / S_1$ dan $\sigma_2 = 0$.
- Untuk *error* = 2, nilai $\sigma_1 = (S_1 S_4 + S_2 S_3) / D_2$ dan $\sigma_2 = (S_2 S_4 + S_3^2) / D_2$

Perhitungan nilai σ tersebut sesuai dengan ketentuan dalam metode Peterson. Diagram blok σ calculator untuk *error* = 1 dan *error* = 2 ditunjukan dalam gambar 3.18.



Gambar 3.18 Diagram blok σ calculator untuk error = 1 dan error = 2

3.4.4. Ω Calculator

Ω *calculator* berfungsi untuk menghitung koefisien *error magnitude polynomial* Ω. Koefisien-koefisien ini digunakan oleh *Chien search* untuk

mendapatkan nilai $\Omega(x)$ yang sesuai dengan lokasi errornya. Untuk persamaan-persamaan berikut :

$$\Omega(x) = S(x)\,\sigma(x) \pmod{2t}$$

$$S(x) = S_1 + S_2\,x + S_3\,x^2 + S_4\,x^3)$$

$$\sigma(x) = \sigma_0 + \sigma_1\,x + \sigma_2\,x^2$$

di mana $\sigma_0 = 1$, koefisien-koefisien persamaan $\Omega(x)$-nya adalah sebagai berikut :

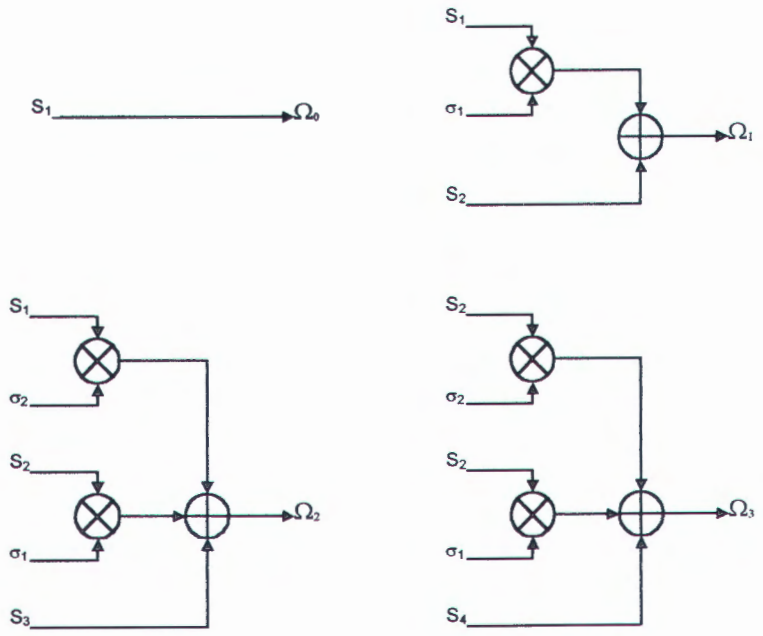
$$\Omega_0 = \sigma_0\ S_1 = S_1$$

$$\Omega_1 = \sigma_0\ S_2 + \sigma_1\ S_1 = S_2 + \sigma_1\ S_1$$

$$\Omega_2 = \sigma_0\ S_3 + \sigma_1\ S_2 + \sigma_2\ S_1 = S_3 + \sigma_1\ S_2 + \sigma_2\ S_1$$

$$\Omega_3 = \sigma_0\ S_4 + \sigma_1\ S_3 + \sigma_2\ S_2 + \sigma_3\ S_1 = S_4 + \sigma_1\ S_3 + \sigma_2\ S_2 + \sigma_3\ S_1$$

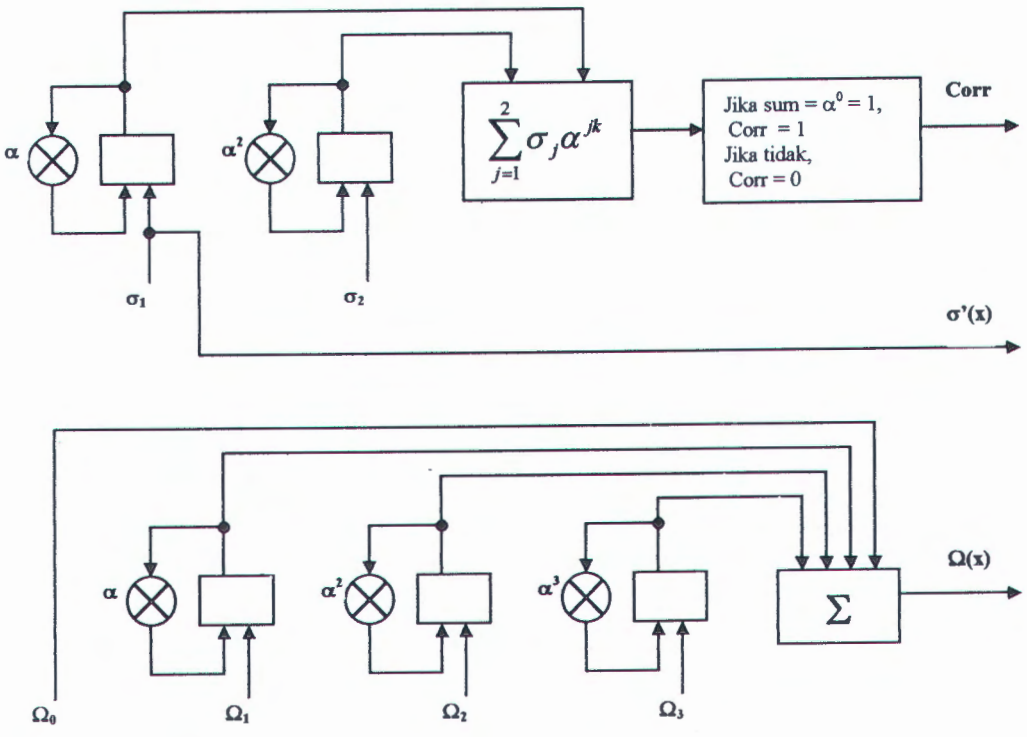
Karena perkaliannya membentuk modulus 4 ($2t = 2.2$), maka koefisien yang dihitung hanya sampai Ω_3 saja. Berdasar persamaan tersebut diagram blok Ω *calculator* ditunjukkan dalam gambar 3.19.



Gambar 3.19 Diagram blok Ω calculator

3.4.5. Chien's Search

Chien search digunakan untuk menunjukkan lokasi error dan menghasilkan nilai turunan σ'(x) dan Ω(x). Nilai σ'(x) dan Ω(x) digunakan algoritma Forney untuk mendapatkan *error value* Y. *Chien search* memeriksa tiap simbol secara berurutan hingga lengkap satu blok (15 simbol). Jika ditemukan adanya *error* pada lokasi tertentu, ia akan mengirim sinyal koreksi Corr. Di saat yang sama, ia mengirimkan nilai σ'(x) dan Ω(x) yang sesuai dengan simbol bersangkutan. Diagram rangkaian *Chien search* untuk t = 2 ditunjukkan dalam gambar 3.20. Setiap jalur dalam gambar 3.20 mempunyai lebar 4 bit.



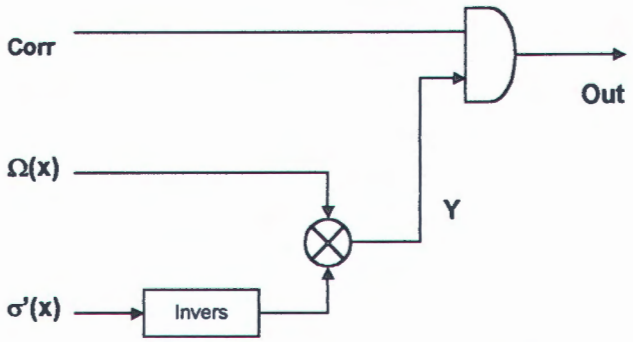
Gambar 3.20 Diagram blok Chien search untuk t = 2

3.4.6. Algoritma Forney dan Error Corrector

Algoritma Forney berfungsi untuk menghitung *error value* Y yang diperoleh dari masukan $\sigma'(x)$ dan $\Omega(x)$ yang sesuai. Persamaan algoritma Forney adalah **(Stanford, 2002)**:

$$Y_i = - \Omega(X_i^{-1}) / \sigma'(X_i^{-1})$$

Maka diagram blok rangkaian algoritma Forney dan *error corrector* seperti ditunjukkan dalam gambar 3.21. Nilai Y hanya dikeluarkan jika sinyal Corr berlogika 1 / aktif.



Gambar 3.21 Diagram blok Forney Algorithm dan Error Corrector

3.4.7. Register FIFO

Register FIFO (*First In First Out*) digunakan untuk menyimpan sementara kode yang masuk, selama proses penghitungan sindrom dan besaran-besaran lain sedang berlangsung. Jika proses penghitungan selesai, isi FIFO dikeluarkan persimbol, sambil diperiksa oleh *Chien search* untuk ditentukan apakah simbol yang bersangkutan mengalami error dan perlu dikoreksi. Panjang register FIFO tergantung pada jumlah langkah yang diperlukan oleh proses *decoding*. Semakin lama proses *decoding*, semakin panjang juga register FIFO yang dibutuhkan. Untuk rangkaian ini,

register FIFO menggunakan *dual port* RAM (DP_RAM) *module* yang tersedia dalam *Logiblock module* FPGA.

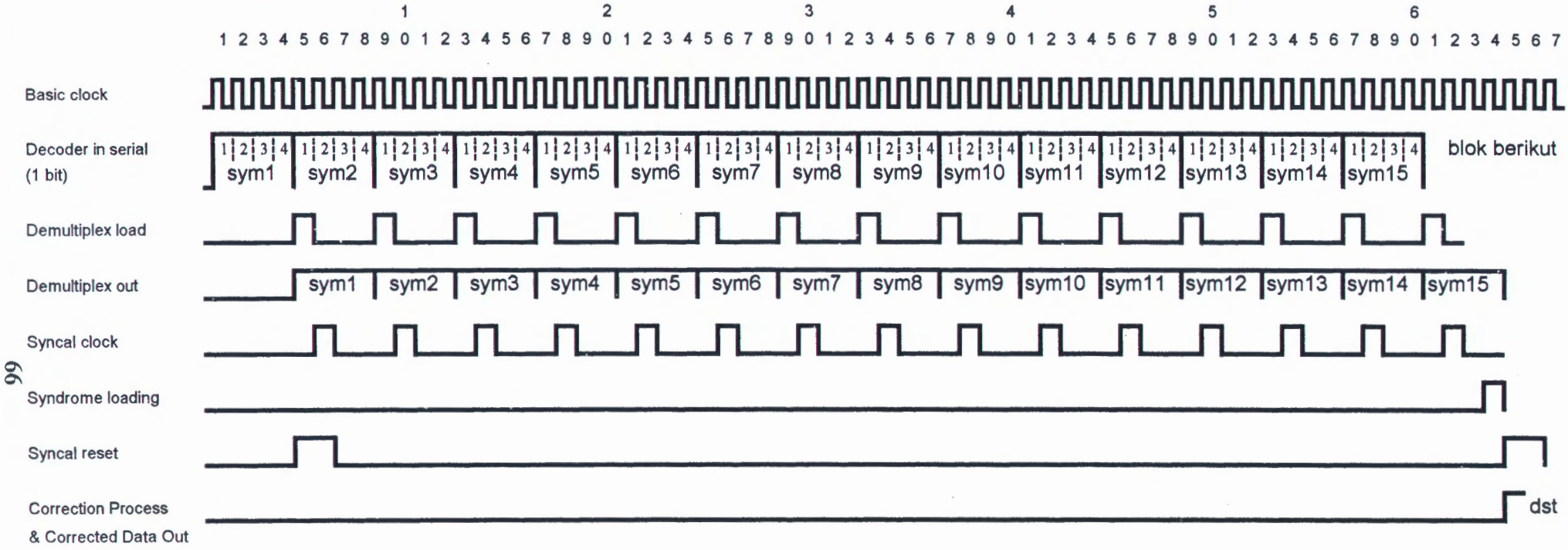
3.4.8. Parity Arbiter

Blok *parity arbiter* berfungsi untuk memisahkan simbol paritas dari simbol informasi, sehingga yang keluar dari blok ini hanya simbol informasi. Kerja *parity arbiter* diatur oleh *counter* yang melakukan hitungan dari 1 hingga 15. Referensi waktunya diambil dari sinyal sinkronisasi. Simbol informasi yang muncul pada clock ke 1 hingga 11, diteruskan ke output *decoder*, sementara simbol paritas yang muncul pada detik ke 12 hingga 15 tidak diteruskan (dibuang).

3.4.9. Perancangan Diagram Waktu

Rancangan diagram waktu proses *decoding* ditunjukkan dalam gambar 3.22. Tampak bahwa tiap simbol diterima dalam 4 satuan waktu dasar (*basic clock*). Hal itu dilakukan untuk mengakomodasi proses demultipleks simbol 4 bit serial menjadi paralel. Demultipleks satu simbol bisa dilakukan jika seluruh bit dari 1 simbol (4 bit) telah diterima. Dengan demikian, proses demultipleks menyebabkan delay proses sebesar 1 simbol. Hal ini ditunjukkan dalam gambar, di mana simbol 1 baru keluar dari multiplexer bersamaan dengan masuknya simbol 2.

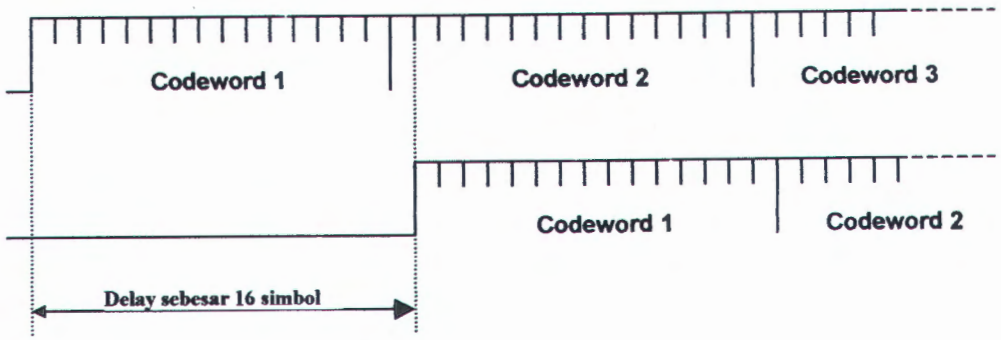
Proses penghitungan sindrom dilaksanakan jika simbol muncul dalam bentuk aslinya (paralel 4 bit). Karena kemunculan simbol pertama terjadi pada saat masuknya simbol serial 2, maka penghitungan sindrom juga mulai terjadi saat itu. Nilai sindrom satu *codeword* diperoleh jika semua simbol *codeword* telah diterima.



Gambar 3.26 Rancangan diagram waktu proses decoding

Dalam diagram waktu tampak bahwa nilai sindrom disimpan ketika nilai simbol terakhir *codeword* telah diterima. Berdasar nilai sindrom inilah, proses koreksi dan pengeluaran data terkoreksi mulai dilakukan.

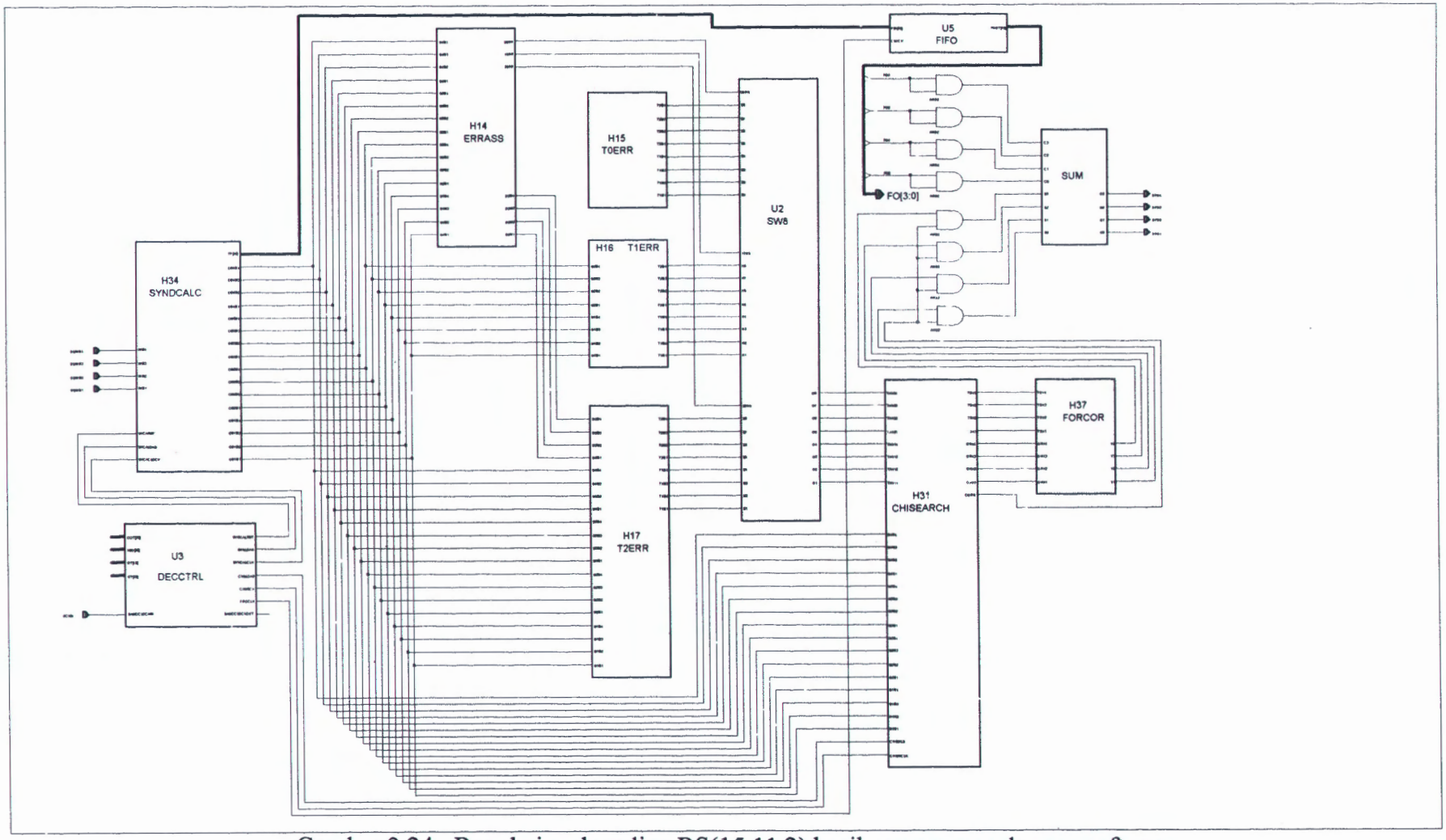
Jadi delay yang terjadi pada data keluaran decoder dibanding data masukan adalah 1 codeword + 1 simbol = 16 simbol. Hal ini ditunjukkan lebih jelas pada gambar 3.23. Tetapi jika data ditransfer secara paralel (4 bit), delay pada keluaran decoder hanya sebesar 1 codeword saja.



Gambar 3.23 Diagram waktu delay antara codeword masukan dan codeword keluaran decoder untuk transfer serial

3.4.10. Membuat Rangkaian Decoder di Software Xilink Foundation F2.1i

Seperti halnya *encoder*, rangkaian *decoder* juga disusun dengan mempertimbangkan karakteristik diagram waktu dari komponen yang dipakai dalam rangkaian (flip-flop D, flip-flop T, gerbang XOR, dan gerbang AND). Penyusunan dilakukan dengan *software* Xilink Foundation Tools. Hasil penyusunan ditunjukkan dalam gambar 3.24. Rangkaian dalam gambar 3.24 merupakan rangkaian dari simbol-simbol makro yang dibentuk oleh gerbang-gerbang logika dasar, flip-flop T , dan flip-flop D.



Gambar 3.24 Rangkaian decoding RS(15,11,2) hasil penyusunan dengan software

3.4.11. Simulasi Decoder RS(15,11,2) dan Pemeriksaan Hasilnya

Proses simulasi dilakukan dengan *software* yang sama. Simulasi dilakukan untuk *error* pada 1 simbol dan *error* pada 2 simbol. Untuk memeriksa kebenaran rangkaian, digunakan satu *codeword* hasil perhitungan proses *decoding* di atas. Persamaan polinomialnya adalah :

$$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$$

Perhitungan *decoding* untuk 1 error dengan koefisien α^{11} di posisi x^2

Jika *error* terjadi di x^2 (simbol ke 13, karena pengiriman dimulai dari pangkat x tertinggi) dengan nilai koefisien α^{11}, maka persamaan polinomial *codeword* yang diterima adalah :

$$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + \boldsymbol{\alpha^{11}} x^2 + \alpha^{11} x + \alpha^{13}$$

Komponen-komponen sindromnya adalah :

$$S_1 = \alpha^{14} + \alpha^{13} + \alpha\,\alpha^{11} + \alpha^2\alpha^{10} + \alpha^3\alpha^9 + \alpha^4\alpha^8 + \alpha^5\alpha^7 + \alpha^6\alpha^6 + \alpha^8\alpha^5 + \alpha^9\alpha^4 + \alpha^{13}\ \alpha^3 + \boldsymbol{\alpha^{11}}\alpha^2 + \alpha^{11}\alpha + \alpha^{13}$$

$$= \alpha^{14} + \alpha^{13} + \alpha^{12} + \alpha^{12} + \alpha^{12} + \alpha^{12} + \alpha^{12} + \alpha^{12} + \alpha^{13} + \alpha^{13} + \alpha^{16} + \alpha^{13} + \alpha^{12} + \alpha^{13}$$

$$= \alpha^{14} + \alpha^{13} + \alpha^{12} + \alpha^{16} = \boldsymbol{\alpha^{14}}$$

$$S_2 = \alpha^{2\cdot 14} + \alpha^{2\cdot 13} + \alpha\,\alpha^{2\cdot 11} + \alpha^2\alpha^{2\cdot 10} + \alpha^3\alpha^{2\cdot 9} + \alpha^4\alpha^{2\cdot 8} + \alpha^5\alpha^{2\cdot 7} + \alpha^6\alpha^{2\cdot 6} + \alpha^8\alpha^{2\cdot 5} + \alpha^9\alpha^{2\cdot 4} + \alpha^{13}\ \alpha^{2\cdot 3} + \boldsymbol{\alpha^{11}}\alpha^{2\cdot 2} + \alpha^{11}\alpha^2 + \alpha^{13}$$

$= \alpha^{28} + \alpha^{26} + \alpha^{23} + \alpha^{22} + \alpha^{21} + \alpha^{20} + \alpha^{19} + \alpha^{18} + \alpha^{18} + \alpha^{17} + \alpha^{19} + \alpha^{15} + \alpha^{13} + \alpha^{13}$

$= \alpha^{13} + \alpha^{11} + \alpha^{8} + \alpha^{7} + \alpha^{6} + \alpha^{5} + \alpha^{2} + 1 = \alpha$

$S_3 = \alpha^{3.14} + \alpha^{3.13} + \alpha\,\alpha^{3.11} + \alpha^{2}\alpha^{3.10} + \alpha^{3}\alpha^{3.9} + \alpha^{4}\alpha^{3.8} + \alpha^{5}\alpha^{3.7} + \alpha^{6}\alpha^{3.6} + \alpha^{8}\alpha^{3.5} + \alpha^{9}\alpha^{3.4} + \alpha^{13}\ \alpha^{3.3} + \alpha^{11}\alpha^{3.2} + \alpha^{11}\alpha^{3} + \alpha^{13}$

$= \alpha^{42} + \alpha^{39} + \alpha^{34} + \alpha^{32} + \alpha^{30} + \alpha^{28} + \alpha^{26} + \alpha^{24} + \alpha^{23} + \alpha^{21} + \alpha^{22} + \alpha^{17} + \alpha^{14} + \alpha^{13}$

$= \alpha^{12} + \alpha^{4} + 1 + \alpha^{11} + \alpha^{8} + \alpha^{6} + \alpha^{7} + \alpha^{14} = \alpha^{3}$

$S_4 = \alpha^{4.14} + \alpha^{4.13} + \alpha\,\alpha^{4.11} + \alpha^{2}\alpha^{4.10} + \alpha^{3}\alpha^{4.9} + \alpha^{4}\alpha^{4.8} + \alpha^{5}\alpha^{4.7} + \alpha^{6}\alpha^{4.6} + \alpha^{8}\alpha^{4.5} + \alpha^{9}\alpha^{4.4} + \alpha^{13}\ \alpha^{4.3} + \alpha^{11}\alpha^{4.2} + \alpha^{11}\alpha^{4} + \alpha^{13}$

$= \alpha^{56} + \alpha^{52} + \alpha^{45} + \alpha^{42} + \alpha^{39} + \alpha^{36} + \alpha^{33} + \alpha^{30} + \alpha^{28} + \alpha^{25} + \alpha^{25} + \alpha^{19} + \alpha^{15} + \alpha^{13}$

$= \alpha^{11} + \alpha^{7} + \alpha^{12} + \alpha^{9} + \alpha^{6} + \alpha^{3} + \alpha^{4} + 1 = \alpha^{5}$

Hingga proses ini, *decoder* belum mengetahui berapa kesalahan yang terjadi. Karena itu, langkah berikutnya adalah mendeteksi jumlah error yang terjadi. Selain itu juga dilakukan penghitungan koefisien polinomial *error locator*. Keduanya menggunakan metode Peterson, dan mempunyai hasil sebagai berikut :

$D_2 = S_1 . S_3 + S_2^{2}$

$= \alpha^{14} . \alpha^{3} + (\alpha)^{2} = 0$

$S_1 = \alpha^{14} \neq 0$

Maka codeword dianggap mengandung 1 error. Sedangkan koefisien polinomial error locatornya adalah :

$\sigma_1 = S_2 / S_1$

$= \alpha / \alpha^{14} = \alpha^2$

Sementara itu nilai $\sigma_2 = 0$, karena codeword hanya mengandung 1 error.

Selanjutnya, dilakukan penghitungan koefisien polinomial error magnitude. Untuk persamaan polinomial sindrom dan polinomial error locator berikut :

$S(x) = S_1 + S_2 x + S_3 x^2 + S_4 x^3 = \alpha^{14} + \alpha x + \alpha^3 x^2 + \alpha^5 x^3$

$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 = 1 + \alpha^2 x$

koefisien-koefisien persamaan $\Omega(x)$-nya adalah :

$\Omega_0 = \sigma_0 S_1 = \alpha^{14}$

$\Omega_1 = \sigma_0 S_2 + \sigma_1 S_1 = \alpha + \alpha^2 \alpha^{14} = 0$

$\Omega_2 = \sigma_0 S_3 + \sigma_1 S_2 + \sigma_2 S_1 = \alpha^3 + \alpha^2 \alpha + 0 = 0$

$\Omega_3 = \sigma_0 S_4 + \sigma_1 S_3 + \sigma_2 S_2 + \sigma_3 S_1 = \alpha^5 + \alpha^2 \alpha^3 + 0 + 0 = 0$

Berikutnya diperlukan juga koefisien dari $\sigma'(x)$ yang merupakan derivatif dari $\sigma(x)$. Untuk persamaan $\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2$, derivatifnya adalah $\sigma'(x) = \sigma_1 + 2 \sigma_2 x^2$. Karena 2 adalah 0 mod 2, maka persamaan derivatifnya menjadi $\sigma'(x) = \sigma_1 = \alpha^2$.

Untuk menemukan lokasi error dan error value yang sesuai dengan lokasinya, digunakan Chien search, yang tiap langkahnya ditunjukkan dalam tabel 3.1. Pada tabel, persamaan $\sigma(x) = \sigma_1 x$, karena $\sigma_2 = 0$. Sedangkan $\Omega(x) = \Omega_0$, karena baik Ω_1, Ω_2 maupun Ω_3 , semuanya bernilai 0. Kemudian dengan algoritma Forney, dapat dihitung error value Y.

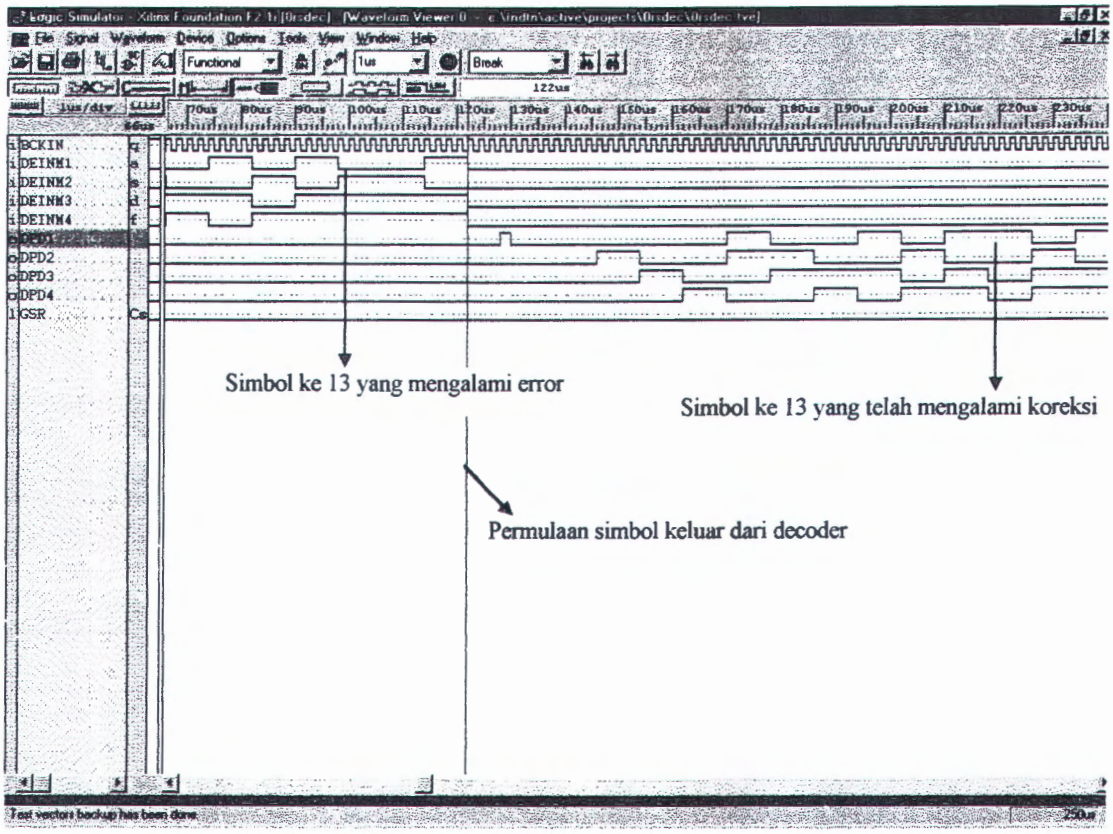
Tabel 3.1 Hasil tiap langkah Chien search untuk koreksi 1 error

x	$\sigma_1 x$	$\sigma(x)$	Status Corr	Ω_0	$\Omega(x)$	$\sigma'(x)$	Y
α	α^3	α^3		α^{14}	α^{14}	α^2	α^{12}
α^2	α^4	α^4		α^{14}	α^{14}	α^2	α^{12}
α^3	α^5	α^5		α^{14}	α^{14}	α^2	α^{12}
α^4	α^6	α^6		α^{14}	α^{14}	α^2	α^{12}
α^5	α^7	α^7		α^{14}	α^{14}	α^2	α^{12}
α^6	α^8	α^8		α^{14}	α^{14}	α^2	α^{12}
α^7	α^9	α^9		α^{14}	α^{14}	α^2	α^{12}
α^8	α^{10}	α^{10}		α^{14}	α^{14}	α^2	α^{12}
α^9	α^{11}	α^{11}		α^{14}	α^{14}	α^2	α^{12}
α^{10}	α^{12}	α^{12}		α^{14}	α^{14}	α^2	α^{12}
α^{11}	α^{13}	α^{13}		α^{14}	α^{14}	α^2	α^{12}
α^{12}	α^{14}	α^{14}		α^{14}	α^{14}	α^2	α^{12}
α^{13}	1	1	Koreksi	α^{14}	α^{14}	α^2	α^{12}
α^{14}	α	α		α^{14}	α^{14}	α^2	α^{12}
1	α^2	α^2		α^{14}	α^{14}	α^2	α^{12}

Berdasar tabel diperoleh bahwa lokasi error berada di simbol ke 13 , dan harus dikoreksi dengan $Y = \alpha^{12}$. Dengan codeword sebelumnya di mana simbol ke 13 adalah α^{11}, koreksi errornya menghasilkan nilai simbol $\alpha^{11} + \alpha^{12} = 1$. Maka kesalahan dapat diperbaiki, dan diperoleh nilai simbol aslinya.

Hasil simulasi decoding 1 error ditunjukkan dalam gambar 3.25. Diagram waktu pada gambar menunjukkan bahwa simbol pertama dikeluarkan bersamaan dengan decoder menerima simbol ke enambelas. Dengan kata lain, kelambatan terjadi sebesar 1 codeword.. Ketidak samaan dua simbol awal dengan simbol asli kemungkinan disebabkan ketidak stabilan kondisi awal pada simulasi. Pada

kenyataannya, setelah dilakukan implementasi dan loading, rangkaian dapat bekerja dengan baik.



Gambar 3.25 Hasil simulasi decoding 1 error

Perhitungan *decoding* untuk 2 error dengan koefisien α^8 di posisi x^{13} dan koefisien α^5 di posisi x^9

Jika *error* terjadi di x^{13} (simbol ke 2) dengan koefisien α^8 dan di x^9 (simbol ke 6) dengan koefisien α^5, maka persamaan polinomial *codeword* yang diterima adalah :

$$c(x) = x^{14} + \alpha^8 x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^5 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$$

Komponen-komponen sindromnya adalah :

$S_1 = \alpha^{14} + \alpha^8 \alpha^{13} + \alpha \alpha^{11} + \alpha^2 \alpha^{10} + \alpha^5 \alpha^9 + \alpha^4 \alpha^8 + \alpha^5 \alpha^7 + \alpha^6 \alpha^6 + \alpha^8 \alpha^5 + \alpha^9 \alpha^4$

$+ \alpha^{13}\ \alpha^3 + \alpha^2 + \alpha^{11} \alpha + \alpha^{13}$

$= \alpha^{14} + \alpha^{21} + \alpha^{12} + \alpha^{12} + \alpha^{14} + \alpha^{12} + \alpha^{12} + \alpha^{12} + \alpha^{13} + \alpha^{13} + \alpha^{16} + \alpha^2 + \alpha^{12} +$

α^{13}

$= \alpha^{21} + \alpha^{16} + \alpha^2 + \alpha^{13} = \boldsymbol{\alpha^{10}}$

$S_2 = \alpha^{2.14} + \alpha^8 \alpha^{2.13} + \alpha \alpha^{2.11} + \alpha^2 \alpha^{2.10} + \alpha^5 \alpha^{2.9} + \alpha^4 \alpha^{2.8} + \alpha^5 \alpha^{2.7} + \alpha^6 \alpha^{2.}$

$^6 + \alpha^8 \alpha^{2.5} + \alpha^9 \alpha^{2.4} + \alpha^{13}\ \alpha^{2.3} + \alpha^{2.2} + \alpha^{11} \alpha^2 + \alpha^{13}$

$= \alpha^{28} + \alpha^{34} + \alpha^{23} + \alpha^{22} + \alpha^{23} + \alpha^{20} + \alpha^{19} + \alpha^{18} + \alpha^{18} + \alpha^{17} + \alpha^{19} + \alpha^4 + \alpha^{13} +$

α^{13}

$= \alpha^{13} + \alpha^7 + \alpha^5 + \alpha^2 = \boldsymbol{\alpha^2}$

$S_3 = \alpha^{3.14} + \alpha^8 \alpha^{3.13} + \alpha \alpha^{3.11} + \alpha^2 \alpha^{3.10} + \alpha^5 \alpha^{3.9} + \alpha^4 \alpha^{3.8} + \alpha^5 \alpha^{3.7} + \alpha^6 \alpha^{3.}$

$^6 + \alpha^8 \alpha^{3.5} + \alpha^9 \alpha^{3.4} + \alpha^{13}\ \alpha^{3.3} + \alpha^{3.2} + \alpha^{11} \alpha^3 + \alpha^{13}$

$= \alpha^{42} + \alpha^{47} + \alpha^{34} + \alpha^{32} + \alpha^{32} + \alpha^{28} + \alpha^{26} + \alpha^{24} + \alpha^{23} + \alpha^{21} + \alpha^{22} + \alpha^6 + \alpha^{14} +$

α^{13}

$= \alpha^{12} + \alpha^2 + \alpha^4 + \alpha^{11} + \alpha^9 + \alpha^8 + \alpha^7 + \alpha^{14} = \boldsymbol{\alpha^7}$

$S_4 = \alpha^{4.14} + \alpha^8 \alpha^{4.13} + \alpha \alpha^{4.11} + \alpha^2 \alpha^{4.10} + \alpha^5 \alpha^{4.9} + \alpha^4 \alpha^{4.8} + \alpha^5 \alpha^{4.7} + \alpha^6 \alpha^{4.}$

$^6 + \alpha^8 \alpha^{4.5} + \alpha^9 \alpha^{4.4} + \alpha^{13}\ \alpha^{4.3} + \alpha^{4.2} + \alpha^{11} \alpha^4 + \alpha^{13}$

$= \alpha^{56} + \alpha^{60} + \alpha^{45} + \alpha^{42} + \alpha^{41} + \alpha^{36} + \alpha^{33} + \alpha^{30} + \alpha^{28} + \alpha^8 + \alpha^{15} + \alpha^{13}$

$= \alpha^{12} + \alpha^6 + \alpha^3 + \alpha^8 = \boldsymbol{\alpha^{11}}$

Hingga proses ini, *decoder* belum mengetahui berapa kesalahan yang terjadi. Karena itu, langkah berikutnya adalah mendeteksi jumlah error yang terjadi. Selain

itu juga dilakukan penghitungan koefisien polinomial *error locator*. Keduanya menggunakan metode Peterson, dan mempunyai hasil sebagai berikut :

$D_2 = S_1 . S_3 + S_2^2$

$= \alpha^{10} . \alpha^7 + (\alpha^2)^2 = \alpha^{10} \neq 0$

$S_1 = \alpha^{14} \neq 0$

Maka codeword dianggap mengandung 2 error, dengan koefisien polinomial error locatornya adalah :

$\sigma_1 = 1/D_2 (S_1 . S_4 + S_2 . S_3)$

$= 1/\alpha^{10} (\alpha^{10} . \alpha^{11} + \alpha^2 . \alpha^7)$

$= \alpha^5 / \alpha^{10} = \alpha^{10}$

$\sigma_2 = 1/D_2 (S_2 . S_4 + S_3^2)$

$= 1/\alpha^{10} (\alpha^2 . \alpha^{11} + \alpha^{7.2})$

$= \alpha^2 / \alpha^{10} = \alpha^7$

Selanjutnya, dilakukan penghitungan koefisien polinomial error magnitude. Untuk persamaan polinomial sindrom dan polinomial error locator berikut :

$S(x) = S_1 + S_2 x + S_3 x^2 + S_4 x^3 = \alpha^{10} + \alpha^2 x + \alpha^7 x^2 + \alpha^{11} x^3$

$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 = 1 + \alpha^{10} x + \alpha^7 x^2$

koefisien-koefisien persamaan $\Omega(x)$-nya adalah :

$\Omega_0 = \sigma_0 S_1 = \alpha^{10}$

$\Omega_1 = \sigma_0 S_2 + \sigma_1 S_1 = \alpha^2 + \alpha^{10} \alpha^{10} = \alpha$

$\Omega_2 = \sigma_0 S_3 + \sigma_1 S_2 + \sigma_2 S_1 = \alpha^7 + \alpha^{10} \alpha^2 + \alpha^7 \alpha^{10} = 0$

$\Omega_3 = \sigma_0 S_4 + \sigma_1 S_3 + \sigma_2 S_2 + \sigma_3 S_1 = \alpha^{11} + \alpha^{10} \alpha^7 + \alpha^7 \alpha^2 + 0 = 0$

Berikutnya diperlukan juga koefisien dari $\sigma'(x)$ yang merupakan derivatif dari $\sigma(x)$. Untuk persamaan $\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2$, derivatifnya adalah $\sigma'(x) = \sigma_1 + 2\sigma_2 x^2$. Karena 2 adalah 0 mod 2, maka persamaan derivatifnya menjadi $\sigma'(x) = \sigma_1 = \alpha^{10}$.

Untuk menemukan lokasi error dan error value yang sesuai dengan lokasinya, digunakan Chien search, yang tiap langkahnya ditunjukkan dalam tabel 3.1. Pada tabel, persamaan $\Omega(x) = \Omega_0 + \Omega_1$, karena Ω_2 maupun Ω_3 , semuanya bernilai 0. Kemudian dengan algoritma Forney, dapat dihitung error value Y.

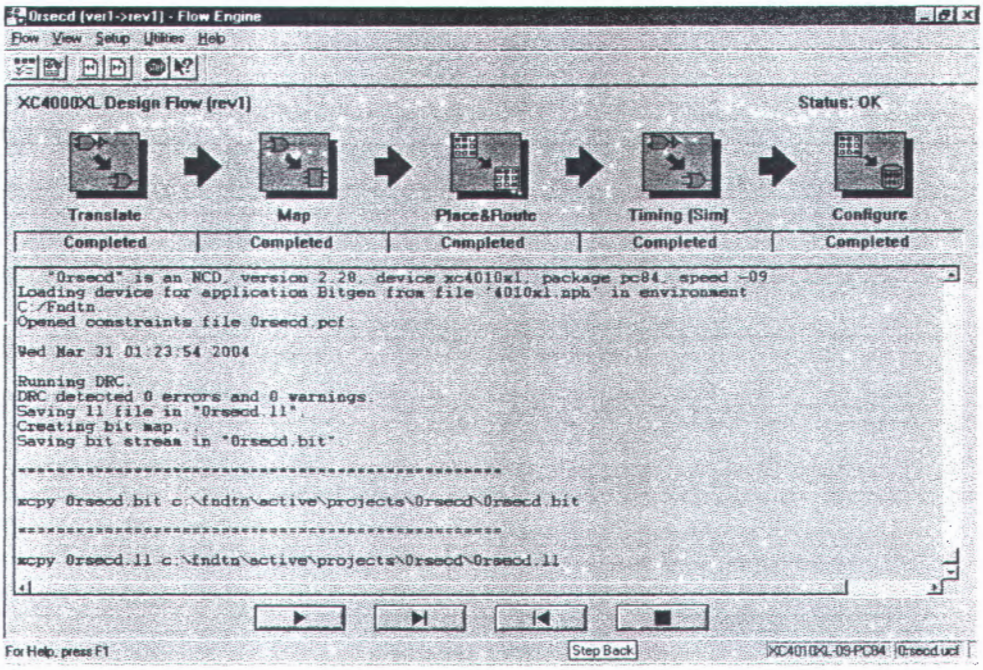
Tabel 3.2 Hasil tiap langkah Chien search untuk koreksi 2 error

X	$\sigma_1 x$	$\sigma_2 x^2$	$\sigma(x)$	Status Corr	Ω_0	Ω_1	$\Omega(x)$	$\sigma'(x)$	Y
α	α^{11}	α^9	α^2		α^{10}	α^2	α^4	α^{10}	α^9
α^2	α^{12}	α^{11}	1	Koreksi	α^{10}	α^3	α^{12}	α^{10}	α^2
α^3	α^{13}	α^{13}	0		α^{10}	α^4	α^2	α^{10}	α^7
α^4	α^{14}	1	α^3		α^{10}	α^5	1	α^{10}	α^5
α^5	1	α^2	α^8		α^{10}	α^6	α^7	α^{10}	α^{12}
α^6	α	α^4	1	Koreksi	α^{10}	α^7	α^6	α^{10}	α^{11}
α^7	α^2	α^6	α^3		α^{10}	α^8	α	α^{10}	α^6
α^8	α^3	α^8	α^{13}		α^{10}	α^9	α^{13}	α^{10}	α^3
α^9	α^4	α^{10}	α^2		α^{10}	α^{10}	0	α^{10}	0
α^{10}	α^5	α^{12}	α^{14}		α^{10}	α^{11}	α^{14}	α^{10}	α^4
α^{11}	α^6	α^{14}	α^8		α^{10}	α^{12}	α^3	α^{10}	α^8
α^{12}	α^7	α	α^{14}		α^{10}	α^{13}	α^9	α^{10}	α^{14}
α^{13}	α^8	α^3	α^{13}		α^{10}	α^{14}	α^{11}	α^{10}	α
α^{14}	α^9	α^5	α^6		α^{10}	1	α^5	α^{10}	α^{10}
1	α^{10}	α^7	α^2		α^{10}	α	α^8	α^{10}	α^{13}

Berdasar tabel diperoleh bahwa lokasi error berada di simbol ke 2 dan 6. Simbol ke 2 harus dikoreksi dengan $Y = \alpha^2$ dan menghasilkan simbol asli dengan nilai $\alpha^8 + \alpha^2 = 1$. Berikutnya, simbol ke 6 dikoreksi dengan $Y = \alpha^{11}$ dan menghasilkan simbol asli dengan nilai $\alpha^5 + \alpha^{11} = \alpha^3$. Maka kesalahan dapat diperbaiki, dan diperoleh nilai simbol aslinya. Hasil simulasi juga menunjukkan kesamaan dengan perhitungan.

3.5. Implementasi Encoder dan Decoder RS(15,11,2)

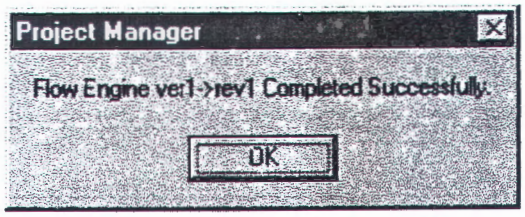
Implementasi *encoder* dan *decoder* RS masih dilakukan dengan *software* Xilink Foundation F2.1i. Sebelum dilakukan implementasi, rangkaian dalam file berekstensi SCH harus dilengkapi dengan IPAD dan IBUF untuk terminal-terminal masukannya, serta OPAD dan OBUF untuk terminal-terminal keluarannya. Tanpa adanya kelengkapan tersebut, implementasi akan mengalami kegagalan. Tampilan program implementasi ditunjukkan dalam gambar 3.26.



Gambar 3.26 Tampilan program Flow Engine implementasi encoder RS

Ketika dilakukan implementasi, secara otomatis *software* akan menjalankan program Flow Engine. Secara berurutan, program akan melakukan proses : *translate, map, place & route, timing simulation*, dan *configure*.

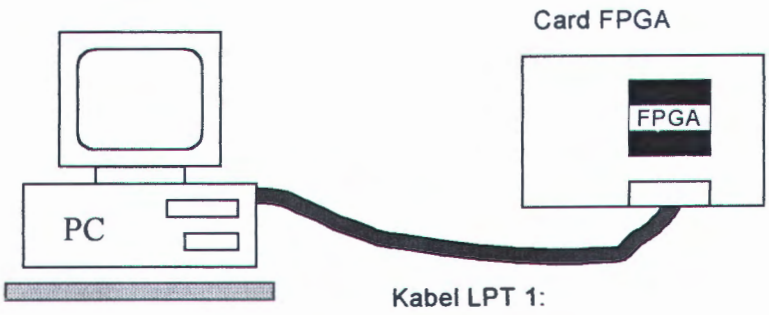
Jika semua proses telah dikerjakan oleh program Flow Engine dan tidak terdapat pesan kesalahan, maka implementasi telah berhasil dilakukan. Indikasi keberhasilan ditunjukkan dalam bentuk pesan seperti dalam gambar 3.27. Hasil akhirnya berupa sebuah *file* dengan ekstensi BIT, yang akan di*load* ke *chip* FPGA.



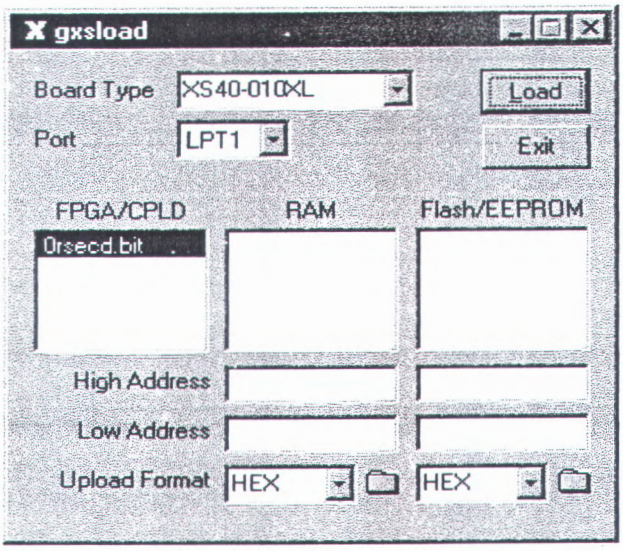
Gambar 3.27 Indikasi keberhasilan implementasi encoder

3.6. Pemrograman/Loading Encoder dan Decoder RS(15,11,2)

Pemrograman/*loading encoder* dan *decoder* RS ke *chip* FPGA XC4010 dilakukan dengan *software* XSTOOLS 4.0.1. Diagram rangkaian *loading*nya ditunjukkan dalam gambar 3.28, dan tampilan programnya ditunjukkan dalam gambar 3.29.



Gambar 3.28 Diagram rangkaian *loading encoder*



Gambar 3.29 Tampilan program *loading encoder*

File yang diload adalah *file* dengan ekstensi BIT hasil implementasi. Proses *loading* berhasil dilakukan jika selama proses tidak terdapat pesan kesalahan, selain itu setelah *loading*, terjadi perubahan pada tampilan LED kit FPGA.

BAB IV

PENGUJIAN CODEC DAN EVALUASI PADA KANAL RADIO

4.1. Pengujian Codec RS

Pengujian *codec* RS dilakukan dalam dua bagian yaitu : pengujian *encoder* dan pengujian *decoder*. Pengujian *encoder* dilakukan dengan memasukkan blok informasi ke *encoder*, kemudian membandingkan *codeword* keluaran dengan *codeword* hasil perhitungan/teoritis.

Pengujian decoder dilakukan dengan memberikan codeword yang mengandung 0 simbol error, 1 simbol error, 2 simbol error, dan 3 simbol error. Kemudian membandingkan hasil perbaikan yang dilakukan oleh *decoder* dengan teori.

4.1.1. Pengujian Encoder

Untuk pengujian encoder, dengan nilai m = 4 bit persimbol dan panjang simbol informasi sebesar 11 simbol, akan diperoleh jumlah kemungkinan blok informasi sebanyak 4^{11} = 4194304 buah. Pengujian dan perhitungan untuk seluruh blok memerlukan waktu dan tenaga yang banyak, sehingga sulit dilaksanakan. Karena itu hanya akan digunakan 10 blok simbol informasi. Tabel 4.1. menunjukkan kesepuluh blok simbol informasi yang akan digunakan dalam pengujian encoder.

Tabel 4.1. Sepuluh blok informasi masukan

Blok ke	Polinomial informasi
1	$x^{10} + x^9 + \alpha x^7 + \alpha^2 x^6 + \alpha^3 x^5 + \alpha^4 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^8 x + \alpha^9$
2	$\alpha^5 x^{10} + \alpha^6 x^9 + \alpha^7 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^{11} x^3 + \alpha^{12} x^2 + \alpha^{13} x + \alpha^{14}$
3	$x^{10} + \alpha x^9 + \alpha^2 x^8 + \alpha^3 x^7 + \alpha^4 x^6 + \alpha^5 x^5 + \alpha^6 x^4 + \alpha^7 x^3 + \alpha^8 x^2 + \alpha^9 x + \alpha^{10}$
4	$\alpha^6 x^4 + \alpha^7 x^3 + \alpha^8 x^2 + \alpha^9 x + \alpha^{10}$
5	$x^{10} + \alpha x^9 + \alpha^2 x^8 + \alpha^3 x^7 + \alpha^4 x^6 + \alpha^5 x^5$
6	$\alpha^{10} x^{10} + \alpha^9 x^9 + \alpha^8 x^8 + \alpha^7 x^7 + \alpha^6 x^6 + \alpha^5 x^5 + \alpha^4 x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha x + 1$
7	$\alpha^9 x^9 + \alpha^7 x^7 + \alpha^5 x^5 + \alpha^3 x^3 + \alpha x$
8	$\alpha^{10} x^{10} + \alpha^8 x^8 + \alpha^6 x^6 + \alpha^4 x^4 + \alpha^2 x^2 + 1$
9	$\alpha x^{10} + \alpha x^9 + \alpha x^8 + \alpha x^7 + \alpha x^6 + \alpha x^5 + \alpha x^4 + \alpha x^3 + \alpha x^2 + \alpha x + \alpha$
10	$x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

Untuk mendapatkan *codeword* teoritis, dilakukan perhitungan dengan urutan seperti telah ditulis dalam subbab 2.5. Untuk blok informasi no 2 pada tabel 4.1, perhitungannya adalah sebagai berikut :

Step 1. Mengalikan informasi u(x) dengan $x^{15-11} = x^4$, dan diperoleh :

$u(x)\, x^4 = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$

Step 2. Mendapatkan paritas p(x) dari sisa pembagian $u(x)\, x^4$ terhadap g(x). Proses perhitungannya adalah sebagai berikut :

$$\alpha^5 x^{10} + \alpha^2 x^9 + \alpha^2 x^8 + \alpha^9 x^7 + \alpha^6 x^6 + x^5 + \alpha^8 x^4 + \alpha^{10} x^3$$

$$x^4 + \alpha^{13} x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10} \;\Big|\; \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$$

$$\alpha^5 x^{14} + \alpha^3 x^{13} + \alpha^{11} x^{12} + \alpha^8 x^{11} + x^{10}$$

$$\alpha^2 x^{13} + \alpha^8 x^{12} + \alpha^{11} x^{11} + \alpha^2 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$$

$$\alpha^2 x^{13} + x^{12} + \alpha^8 x^{11} + \alpha^5 x^{10} + \alpha^{12} x^9$$

$$\alpha^2 x^{12} + \alpha^7 x^{11} + \alpha\, x^{10} + \alpha^8 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$$

$$\alpha^2 x^{12} + x^{11} + \alpha^8 x^{10} + \alpha^5 x^9 + \alpha^{12} x^9$$

$$\alpha^9 x^{11} + \alpha^{10} x^{10} + \alpha^4 x^9 + \alpha^3 x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$$

$$\alpha^9 x^{11} + \alpha^7 x^{10} + x^9 + \alpha^{12} x^8 + \alpha^4 x^7$$

$$\alpha^6 x^{10} + \alpha\, x^9 + \alpha^{10} x^8 + \alpha^{13} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$$

$$\alpha^6 x^{10} + \alpha^4 x^9 + \alpha^{12} x^8 + \alpha^9 x^7 + \alpha\, x^6$$

$$x^9 + \alpha^3 x^8 + \alpha^{10} x^7 + \alpha^{13} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4$$

$$x^9 + \alpha^{13} x^8 + \alpha^6 x^7 + \alpha^3 x^6 + \alpha^{10} x^5$$

$$\alpha^8 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{14} x^4$$

$$\alpha^8 x^8 + \alpha^6 x^7 + \alpha^{14} x^6 + \alpha^{11} x^5 + \alpha^3 x^4$$

$$\alpha^{10} x^7 + \alpha^6 x^6 + \alpha^2 x^5 + x^4$$

$$\alpha^{10} x^7 + \alpha^8 x^6 + \alpha x^5 + \alpha^{13} x^4 + \alpha^5 x^3$$

$$\alpha^{14} x^6 + \alpha^5 x^5 + \alpha^6 x^4 + \alpha^5 x^3$$

$$\alpha^{14} x^6 + \alpha^{12} x^5 + \alpha^5 x^4 + \alpha^2 x^3 + \alpha^9 x^2$$

$$\alpha^{14} x^5 + \alpha^9 x^4 + \alpha\, x^3 + \alpha^9 x^2$$

$$\alpha^{14} x^5 + \alpha^{12} x^4 + \alpha^5 x^3 + \alpha^2 x^2 + \alpha^9 x$$

$$\alpha^8 x^4 + \alpha^2 x^3 + \alpha^{11} x^2 + \alpha^9 x$$

$$\alpha^8 x^4 + \alpha^6 x^3 + \alpha^{14} x^2 + \alpha^{11} x + \alpha^3$$

$$\alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$$

diperoleh $p(x) = \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$

Step 3. Mendapatkan polinomial *codeword* dengan menggabungkan x^{n-k} u(x) dan $p(x) \Rightarrow c(x) = x^{n-k}\, u(x) + p(x)$, dan hasilnya adalah :

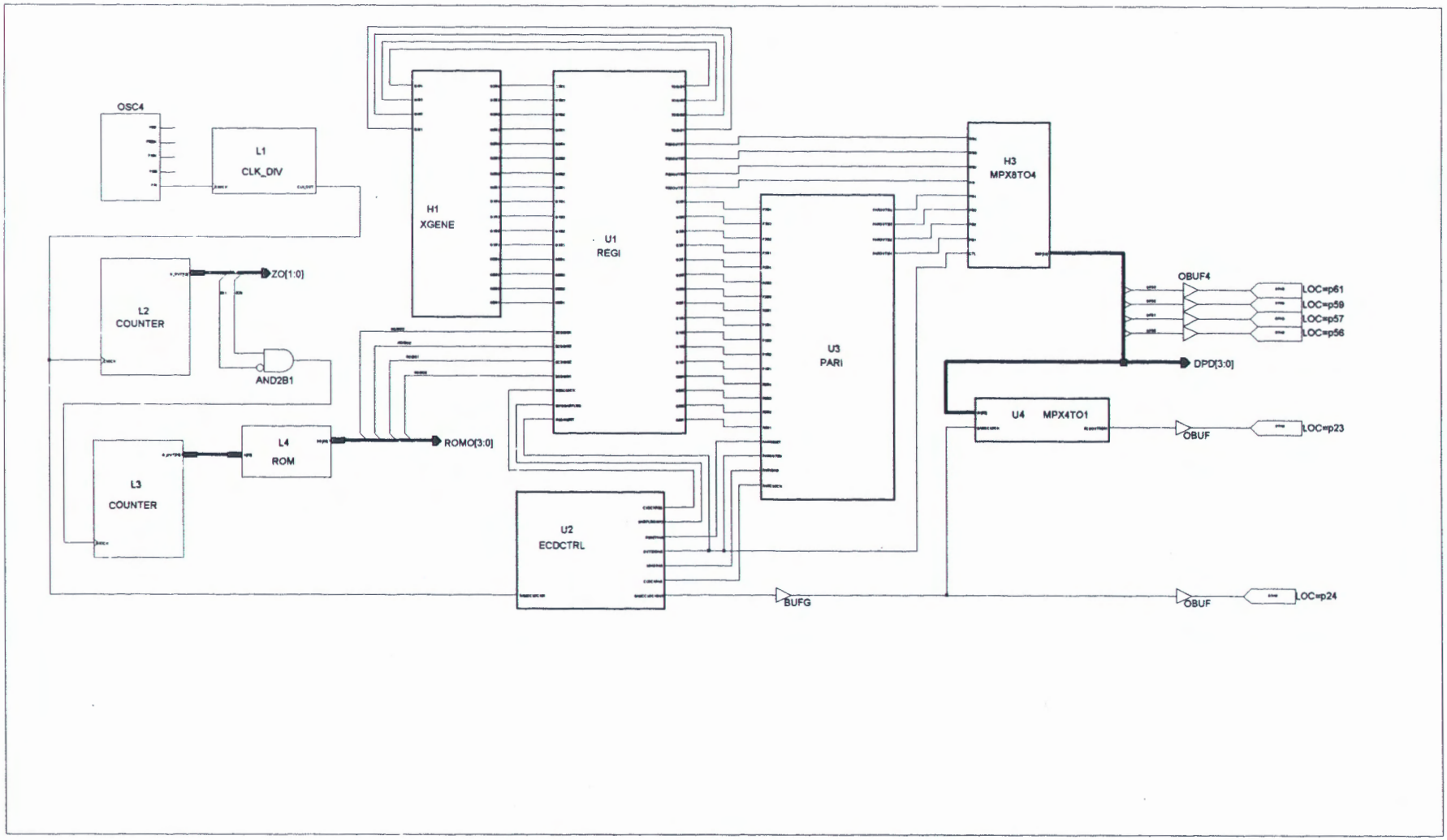
$$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$$

Dengan cara yang sama, tiap-tiap blok informasi dalam tabel 4.1. mempunyai codeword seperti ditunjukkan dalam tabel 4.2.

Tabel 4.2 Codeword hasil perhitungan untuk blok informasi

Blok ke	Polinomial informasi dan polinomial codeword-nya
1	$u(x) = x^{10} + x^9 + \alpha x^7 + \alpha^2 x^6 + \alpha^3 x^5 + \alpha^4 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^8 x + \alpha^9$ $c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
2	$u(x) = \alpha^5 x^{10} + \alpha^6 x^9 + \alpha^7 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^{11} x^3 + \alpha^{12} x^2 + \alpha^{13} x + \alpha^{14}$ $c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
3	$u(x) = x^{10} + \alpha x^9 + \alpha^2 x^8 + \alpha^3 x^7 + \alpha^4 x^6 + \alpha^5 x^5 + \alpha^6 x^4 + \alpha^7 x^3 + \alpha^8 x^2 + \alpha^9 x + \alpha^{10}$ $c(x) = x^{14} + \alpha x^{13} + \alpha^2 x^{12} + \alpha^3 x^{11} + \alpha^4 x^{10} + \alpha^5 x^9 + \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^{12} x^3 + \alpha^6 x + \alpha^4$
4	$u(x) = \alpha^6 x^4 + \alpha^7 x^3 + \alpha^8 x^2 + \alpha^9 x + \alpha^{10}$ $c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$
5	$u(x) = x^{10} + \alpha x^9 + \alpha^2 x^8 + \alpha^3 x^7 + \alpha^4 x^6 + \alpha^5 x^5$ $c(x) = x^{14} + \alpha x^{13} + \alpha^2 x^{12} + \alpha^3 x^{11} + \alpha^4 x^{10} + \alpha^5 x^9 + \alpha^{14} x^3 + \alpha^{13} x^2 + \alpha^{14} x + \alpha^2$
6	$u(x) = \alpha^{10} x^{10} + \alpha^9 x^9 + \alpha^8 x^8 + \alpha^7 x^7 + \alpha^6 x^6 + \alpha^5 x^5 + \alpha^4 x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha x + 1$ $c(x) = \alpha^{10} x^{14} + \alpha^9 x^{13} + \alpha^8 x^{12} + \alpha^7 x^{11} + \alpha^6 x^{10} + \alpha^5 x^9 + \alpha^4 x^8 + \alpha^3 x^7 + \alpha^2 x^6 + \alpha x^5 + x^4 + \alpha^{14} x^3 + \alpha^{13} x^2 + \alpha^{12} x + \alpha^{11}$
7	$u(x) = \alpha^9 x^9 + \alpha^7 x^7 + \alpha^5 x^5 + \alpha^3 x^3 + \alpha x$ $c(x) = \alpha^9 x^{13} + \alpha^7 x^{11} + \alpha^5 x^9 + \alpha^3 x^7 + \alpha x^5 + \alpha^{10} x^3 + \alpha^{13} x^2 + \alpha^2 x + \alpha^8$
8	$u(x) = \alpha^{10} x^{10} + \alpha^8 x^8 + \alpha^6 x^6 + \alpha^4 x^4 + \alpha^2 x^2 + 1$ $c(x) = \alpha^{10} x^{14} + \alpha^8 x^{12} + \alpha^6 x^{10} + \alpha^4 x^8 + \alpha^2 x^6 + x^4 + \alpha^{11} x^3 + \alpha^7 x + \alpha^7$
9	$u(x) = \alpha x^{10} + \alpha x^9 + \alpha x^8 + \alpha x^7 + \alpha x^6 + \alpha x^5 + \alpha x^4 + \alpha x^3 + \alpha x^2 + \alpha x + \alpha$ $c(x) = \alpha x^{14} + \alpha x^{13} + \alpha x^{12} + \alpha x^{11} + \alpha x^{10} + \alpha x^9 + \alpha x^8 + \alpha x^7 + \alpha x^6 + \alpha x^5 + \alpha x^4 + \alpha x^3 + \alpha x^2 + \alpha x + \alpha$
10	$u(x) = x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ $c(x) = x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

Rangkaian pengujian *encoder* ditunjukkan dalam gambar 4.1. Informasi di kirim oleh ROM internal FPGA. Informasi dan *codeword* yang dihasilkan oleh



Gambar 4.1 Rangkaian pengujian encoder

encoder ditampilkan pada *LED*. Frekuensi pulsa clock diturunkan agar, *codeword* keluaran encoder dapat diamati dengan seksama.

Pengujian pada *encoder* dilakukan dengan mengubah konstanta polinomial informasi menjadi bentuk *tuple* 4 bit. Bentuk *tuple* inilah yang dikirim oleh ROM internal FPGA ke masukan *encoder*. Kemudian, dilakukan pengamatan sinyal yang keluar dari *port* P56, P57, P59, dan P61. Sinyal keluaran ini merupakan *codeword* dari informasi yang dimasukkan.

Agar dapat dibandingkan dengan polinomial *codeword* perhitungan, codeword hasil pengamatan (yang masih berupa sinyal digital) harus diubah kembali ke bentuk polinomial biner. Perbandingan polinomial *codeword* hasil perhitungan dan pengamatan ditunjukkan dalam tabel 4.3. Tampak bahwa kesepuluh codeword yang dihasilkan *encoder* mempunyai simbol yang sama dengan codeword perhitungan. Maka dapat dinyatakan bahwa rangkaian encoder telah bekerja dengan benar.

Tabel 4.3 Perbandingan codeword hasil perhitungan dan pengamatan

Blok ke	Polinomial codeword perhitungan dan pengamatan
1	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$ (perhitungan) **$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$ (pengamatan)**
2	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$ (perhitungan) **$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$ (pengamatan)**

Tabel 4.3 Perbandingan codeword hasil perhitungan dan pengamatan (lanjutan)

Blok ke	Polinomial codeword perhitungan dan pengamatan
3	$c(x) = x^{14} + \alpha x^{13} + \alpha^2 x^{12} + \alpha^3 x^{11} + \alpha^4 x^{10} + \alpha^5 x^9 + \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^{12} x^3 + \alpha^6 x + \alpha^4$ (perhitungan) **$c(x) = x^{14} + \alpha x^{13} + \alpha^2 x^{12} + \alpha^3 x^{11} + \alpha^4 x^{10} + \alpha^5 x^9 + \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^{12} x^3 + \alpha^6 x + \alpha^4$ (pengamatan)**
4	$c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$ (perhitungan) **$c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$ (pengamatan)**
5	$c(x) = x^{14} + \alpha x^{13} + \alpha^2 x^{12} + \alpha^3 x^{11} + \alpha^4 x^{10} + \alpha^5 x^9 + \alpha^{14} x^3 + \alpha^{13} x^2 + \alpha^{14} x + \alpha^2$ (perhitungan) **$c(x) = x^{14} + \alpha x^{13} + \alpha^2 x^{12} + \alpha^3 x^{11} + \alpha^4 x^{10} + \alpha^5 x^9 + \alpha^{14} x^3 + \alpha^{13} x^2 + \alpha^{14} x + \alpha^2$ (pengamatan)**
6	$c(x) = \alpha^{10} x^{14} + \alpha^9 x^{13} + \alpha^8 x^{12} + \alpha^7 x^{11} + \alpha^6 x^{10} + \alpha^5 x^9 + \alpha^4 x^8 + \alpha^3 x^7 + \alpha^2 x^6 + \alpha x^5 + x^4 + \alpha^{14} x^3 + \alpha^{13} x^2 + \alpha^{12} x + \alpha^{11}$ (perhitungan) **$c(x) = \alpha^{10} x^{14} + \alpha^9 x^{13} + \alpha^8 x^{12} + \alpha^7 x^{11} + \alpha^6 x^{10} + \alpha^5 x^9 + \alpha^4 x^8 + \alpha^3 x^7 + \alpha^2 x^6 + \alpha x^5 + x^4 + \alpha^{14} x^3 + \alpha^{13} x^2 + \alpha^{12} x + \alpha^{11}$ (pengamatan)**
7	$c(x) = \alpha^9 x^{13} + \alpha^7 x^{11} + \alpha^5 x^9 + \alpha^3 x^7 + \alpha x^5 + \alpha^{10} x^3 + \alpha^{13} x^2 + \alpha^2 x + \alpha^8$ (perhitungan) **$c(x) = \alpha^9 x^{13} + \alpha^7 x^{11} + \alpha^5 x^9 + \alpha^3 x^7 + \alpha x^5 + \alpha^{10} x^3 + \alpha^{13} x^2 + \alpha^2 x + \alpha^8$ (pengamatan)**
8	$c(x) = \alpha^{10} x^{14} + \alpha^8 x^{12} + \alpha^6 x^{10} + \alpha^4 x^8 + \alpha^2 x^6 + x^4 + \alpha^{11} x^3 + \alpha^7 x + \alpha^7$ (perhitungan) **$c(x) = \alpha^{10} x^{14} + \alpha^8 x^{12} + \alpha^6 x^{10} + \alpha^4 x^8 + \alpha^2 x^6 + x^4 + \alpha^{11} x^3 + \alpha^7 x + \alpha^7$ (pengamatan)**

Tabel 4.3 Perbandingan codeword hasil perhitungan dan pengamatan (lanjutan)

Blok ke	Polinomial codeword perhitungan dan pengamatan
9	$c(x) = \alpha x^{14} + \alpha x^{13} + \alpha x^{12} + \alpha x^{11} + \alpha x^{10} + \alpha x^9 + \alpha x^8 + \alpha x^7 + \alpha x^6 + \alpha x^5 + \alpha x^4 + \alpha x^3 + \alpha x^2 + \alpha x + \alpha$ (perhitungan) **$c(x) = \alpha x^{14} + \alpha x^{13} + \alpha x^{12} + \alpha x^{11} + \alpha x^{10} + \alpha x^9 + \alpha x^8 + \alpha x^7 + \alpha x^6 + \alpha x^5 + \alpha x^4 + \alpha x^3 + \alpha x^2 + \alpha x + \alpha$ (pengamatan)**
10	$c(x) = x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ (perhitungan) **$c(x) = x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ (pengamatan)**

4.1.2. Pengujian Decoder

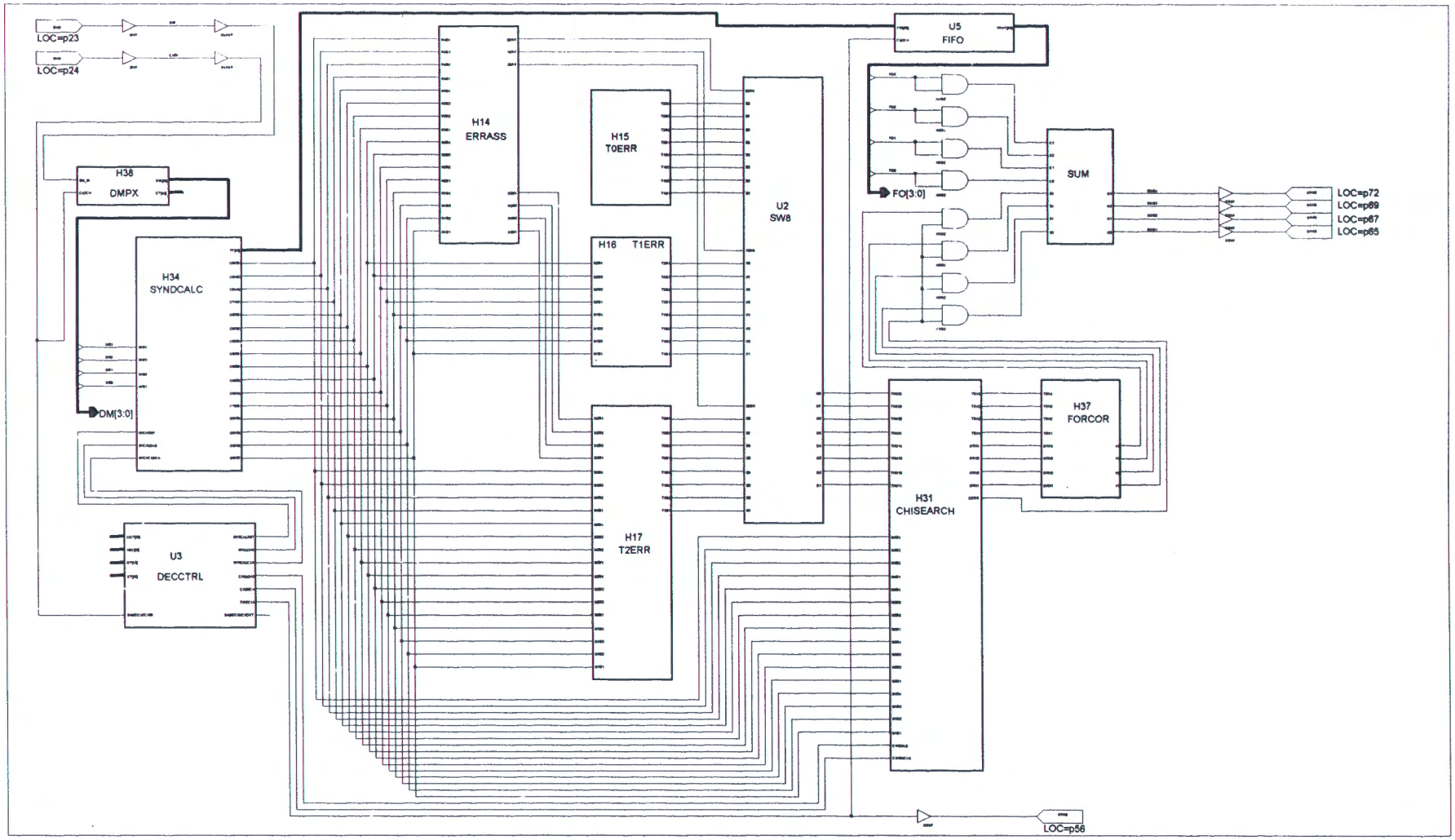
Pengujian *decoder* dilakukan dalam dua bagian, yaitu pengujian terhadap kesalahan simbol dan pengujian terhadap kesalahan akibat *burst (burst error)*. Pengujian terhadap kesalahan simbol dilakukan dalam empat kondisi, yaitu dengan kesalahan sebanyak: 0, 1, 2 dan 3 simbol. Pemilihan ini sesuai dengan nilai kemampuan koreksi maksimal sebanyak 2. Kesalahan 3 simbol digunakan untuk membuktikan bahwa codeword yang masuk dengan kesalahan 3 simbol tersebut tidak dapat diperbaiki.

Pengujian terhadap kesalahan akibat *burst* dilakukan untuk membuktikan bahwa walaupun data berbentuk serial yang terkirim mengalami kesalahan bit beruntun akibat gangguan burst, informasi awal tetap dapat diperbaiki sepanjang deretan bit errornya tidak mencapai lebih dari 2 simbol. Pengujian ini dilakukan

dalam tiga kondisi, yaitu: deretan error perbit yang mencapai 1 simbol, 2 simbol dan 3 simbol. Kesalahan burst dengan deretan error bit mencapai 3 simbol, digunakan untuk menunjukkan bahwa error tidak dapat diperbaiki, karena merusak lebih dari 2 simbol.

Rangkaian decoder yang akan diuji, ditunjukkan dalam gambar 4.2. Tampak bahwa rangkaiannya sama dengan decoder yang telah diuji dalam simulasi, tetapi mendapat tambahan rangkaian demultiplexer. Fungsi demultiplexer adalah untuk mengembalikan bentuk serial data menjadi bentuk paralel 4 bitnya kembali. Simbol masukan decoder dapat diamati di port : P56, P57, P59 dan P61. Sedang simbol keluaran decoder (yang telah mengalami proses decoding) dapat diamati di port : P65, P67, P69, dan P72.

Hasil pengujian akan menunjukkan apakah decoder dapat berfungsi sesuai dengan ketentuan dalam teori. Jika decoder berfungsi dengan benar, maka saat dievaluasi di kanal radio, decoder ditempatkan pada sisi penerima. Letaknya antara demodulator dengan penerima data.

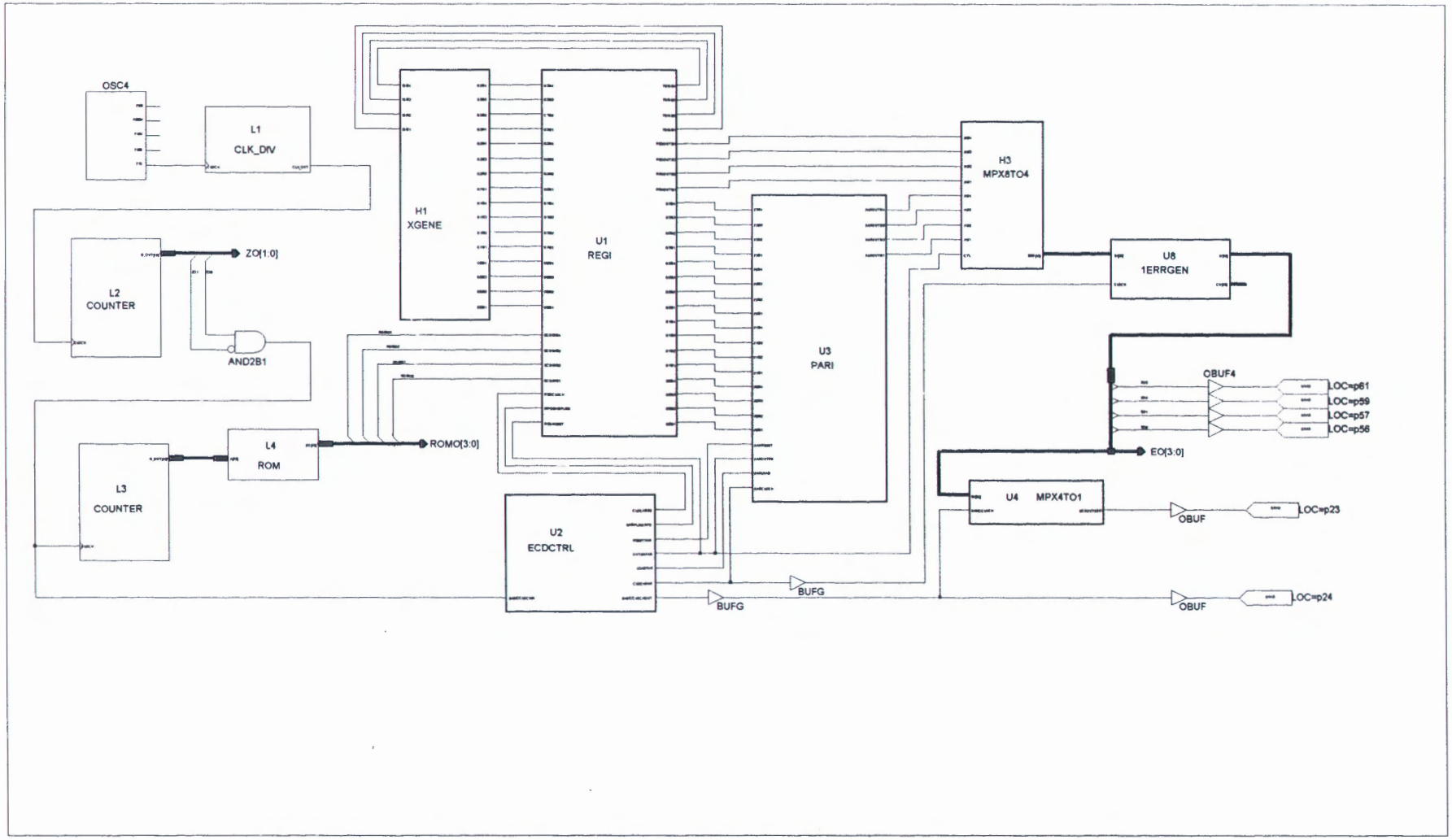


Gambar 4.2 Rangkaian decoder yang akan diuji

4.1.2.1. Pengujian Terhadap Kesalahan Simbol

Pengujian kesalahan simbol dilakukan dengan menambahkan blok generator simbol *error* 1ERRGEN, 2ERRGEN, atau 3ERRGEN pada keluaran *encoder*. Rangkaian encoder untuk pengujian *decoder* dengan kesalahan 1 simbol ditunjukkan dalam gambar 4.3. Generator simbol *error* dipasang pada masukan multiplexer MPX4TO1. Penempatan generator simbol error ditempat itu untuk meyakinkan bahwa *error* yang terjadi adalah simbol *error* yang dikehendaki. Keluaran multiplexer dihubungkan ke masukan decoder. *Codeword* yang telah melalui *decoder* ditampilkan oleh LED. Frekuensi pulsa clock diturunkan, agar keluaran decoder dapat diamati dengan seksama.

Pengujian dilakukan dengan memberikan sebuah codeword dalam empat kondisi, yaitu : tanpa simbol error, dengan 1 simbol error, dengan 2 simbol error, dengan 3 simbol error. Lokasi dan nilai simbol *error* bersifat sembarang, yang penting memenuhi jumlah simbol *error* di atas dan memenuhi nilai simbol *error* yang mungkin terjadi. Karena sebuah *codeword* dicoba dalam empat kondisi, maka pengujian hanya dilakukan untuk tiga *codeword* saja. Daftar *codeword* (dengan empat kondisinya) yang akan dimasukkan ke *decoder* ditunjukkan dalam tabel 4.4. Konstanta/simbol dalam tampilan tebal (bold) merupakan nilai error simbol yang terjadi pada posisi yang bersangkutan. Setelah dilakukan pengujian, *codeword* keluaran *decoder* dibandingkan dengan *codeword* yang diterima. Hasilnya ditunjukkan dalam tabel 4.5.



Gambar 4.3 Rangkaian encoder untuk pengujian decoder terhadap kesalahan 1 simbol

Tabel 4.4 Daftar 3 codeword masukan decoder dengan 4 kondisinya

Blok ke	Polinomial 3 codeword dengan 4 kondisinya
1	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$ (0 simbol error) $c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + \alpha^{11} x^2 + \alpha^{11} x + \alpha^{13}$ (1 simbol error) $c(x) = x^{14} + \alpha^8 x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^5 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$ (2 simbol error) $c(x) = x^{14} + x^{13} + \alpha^3 x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^5 x^8 + \alpha^7 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$ (3 simbol error)
2	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$ (0 simbol error) $c(x) = \alpha^7 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$ (1 simbol error) $c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha x^{12} + \alpha^7 x^{11} + \alpha^9 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$ (2 simbol error) $c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^2 x^6 + \alpha^{11} x^5 + \alpha^{14} x^4 + 1x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$ (3 simbol error)
3	$c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$ (0 simbol error) $c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^4 x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$ (1 simbol error) $c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^3 x + \alpha^{11}$ (2 simbol error) $c(x) = \alpha x^8 + \alpha x^7 + \alpha x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$ (3 simbol error)

Tabel 4.5 Hasil pengujian decoder terhadap simbol error

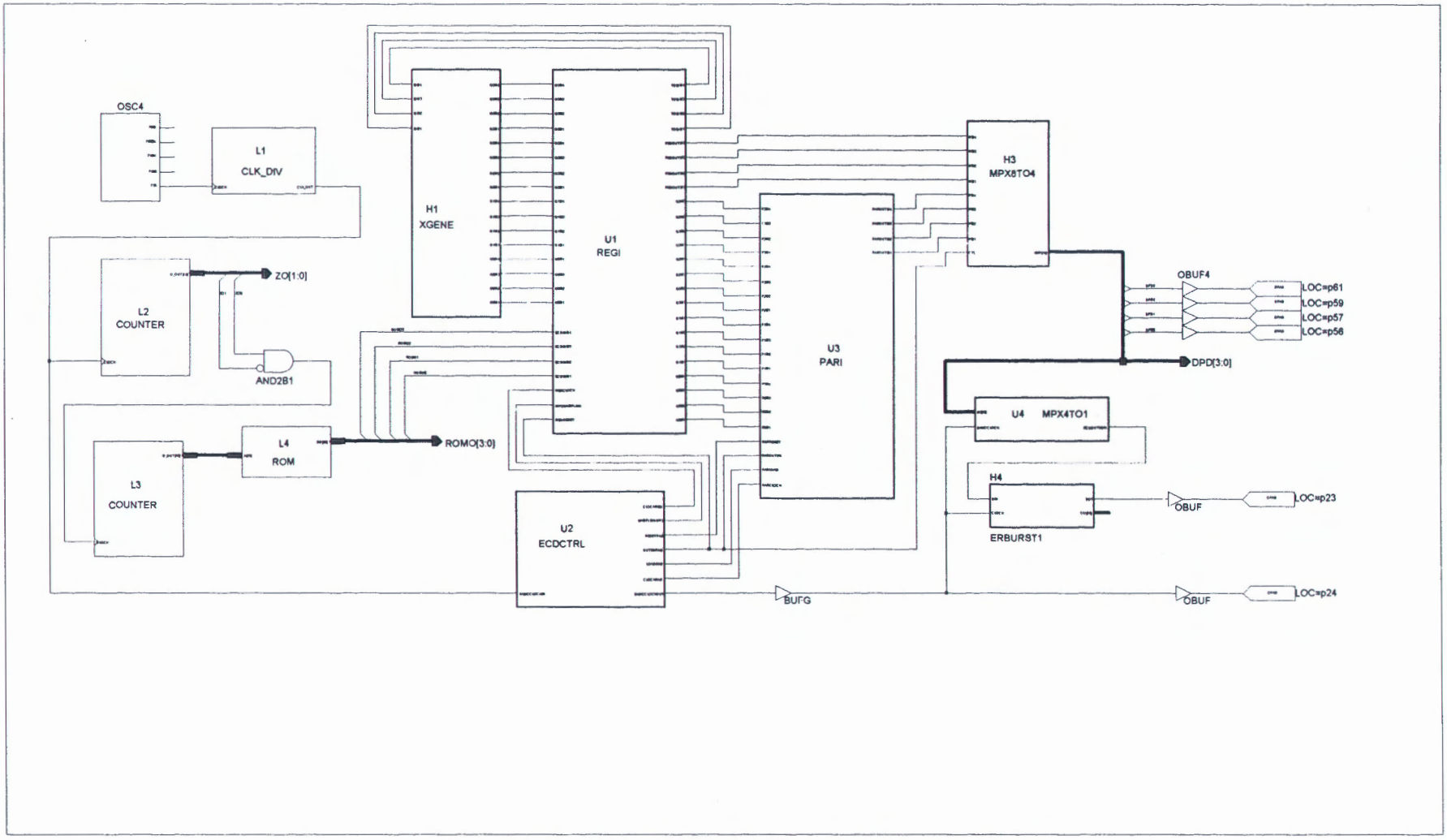
Blok ke	Simbol Error	Polinomial 3 codeword hasil decoding
1	0	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	1	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	2	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	3	$c(x) = x^{14} + x^{13} + \mathbf{\alpha^3} x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \mathbf{\alpha^2} x^8 + \mathbf{\alpha^{14}} x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \mathbf{\alpha^3} x^4 + \alpha^{13} x^3 + x^2 + \mathbf{\alpha^6} x + \alpha^{13}$
2	0	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
	1	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
	2	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
	3	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \mathbf{\alpha^5} x^{11} + \mathbf{\alpha^5} x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \mathbf{\alpha^2} x^6 + \mathbf{\alpha^{11}} x^5 + \alpha^{14} x^4 + \mathbf{1} x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
3	0	$c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$
	1	$c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$
	2	$c(x) = \alpha^6 x^8 + \alpha^7 x^7 + \alpha^8 x^6 + \alpha^9 x^5 + \alpha^{10} x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^8 x + \alpha^{10}$
	3	$c(x) = \mathbf{\alpha^2} x^8 + \mathbf{\alpha^2} x^7 + \mathbf{\alpha^2} x^6 + \mathbf{\alpha^2} x^5 + \mathbf{\alpha^{12}} x^4 + \alpha^6 x^3 + \mathbf{\alpha^9} x^2 + \alpha^7 x + \mathbf{\alpha^{11}}$

Dengan membandingkan isi tabel 4.4. dan tabel 4.5, tampak bahwa *codeword* yang mengalami 1 dan 2 simbol *error* telah diperbaiki dan sama dengan *codeword* asal (dengan 0 simbol *error*). *Codeword* dengan 3 simbol *error* tidak dapat dikoreksi, bahkan kesalahan simbol semakin bertambah. Codeword 1 bertambah 2 kesalahan (menjadi 5 kesalahan), codeword 2 bertambah 2 kesalahan (menjadi 5 kesalahan), dan codeword 3 bertambah 6 kesalahan (menjadi 9 kesalahan). Kenyataan ini sesuai dengan kemampuan koreksi maksimal kode RS yang sebesar $t = 2$

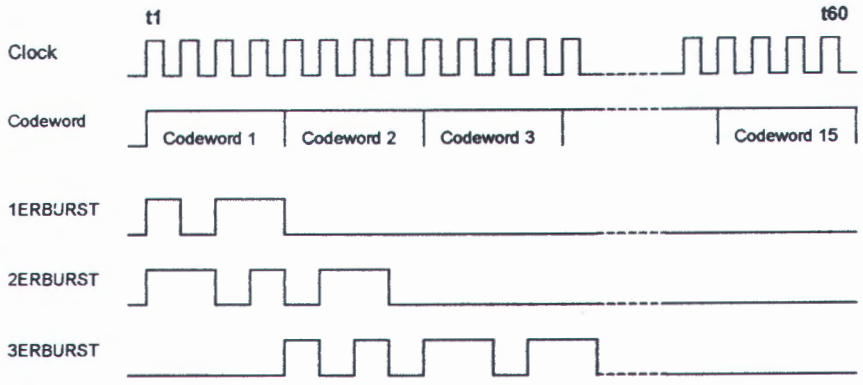
4.1.2.2. Pengujian Terhadap Kesalahan Akibat Burst

Pengujian terhadap kesalahan akibat burst dilakukan dengan menambahkan blok generator error burst ERBURST1, ERBURST2, dan ERBURST3 pada keluaran encoder. Tepatnya pada keluaran blok multiplekser MPX4TO1, seperti ditunjukkan dalam gambar 4.4.

ERBURST1 adalah generator error burst yang menyebabkan rusaknya 1 simbol error, ERBURST2 menyebabkan rusaknya 2 simbol error, sedang ERBURST3 menyebabkan rusaknya 3 simbol error. Gambar 4.5. menunjukkan pola error burst yang dihasilkan oleh ketiga generator error burst. Pola ini disesuaikan dengan codeword berbentuk serial yang satu simbolnya dikeluarkan dalam 4 pulsa clock, dan satu codeword dikirim dalam 60 pulsa clock.



Gambar 4.4 Rangkaian encoder pengujian error burst



Gambar 4.5. Pola error burst yang dihasilkan oleh tiga generator

Penempatan generator di keluaran blok MPX4TO1 bertujuan untuk mendapatkan pengaruh burst yang mempengaruhi bentuk codeword yang dikirim dalam bentuk serial. *Codeword* yang terjadi akibat burst diamati pada keluaran DMPX decoder (Port P56, P57, P59, P61) dalam gambar 4.2. Sementara codeword hasil proses decoding diamati pada keluaran port P65, P67, P69, dan P72. Jumlah dan persamaan codeword yang dimasukkan, sama dengan pengujian decoder terhadap kesalahan simbol. Hasil pengujian decoder terhadap adanya error burst ditunjukkan dalam tabel 4.6.

Dalam tabel 4.6. tampak bahwa kesalahan sebesar 1 simbol (diakibatkan 1ERBURST) dan sebesar 2 simbol (diakibatkan 2ERBURST) masih mampu diperbaiki decoder, karena hanya merusak maksimal 2 simbol. Ini sesuai dengan kemampuan koreksi maksimal kode yang sebesar t = 2. Sementara kesalahan yang diakibatkan adanya burst perusak 3 simbol (yang dibangkitkan 3ERBURST) tidak dapat diperbaiki oleh decoder. Bahkan jumlah kesalahan menjadi bertambah. Hal itu

dapat dilihat pada baris : error burst **3 decoder out**, ketika dibandingkan dengan baris **3 decoder in** di ketiga codeword yang diuji.

Tabel 4.6. Hasil pengujian decoder terhadap error burst

Blok ke	Error Burst	Polinomial 3 codeword
1	Data asal	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	1 decoder in	$c(x) = \alpha^6 x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	1 decoder out	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	2 decoder in	$c(x) = \alpha^9 x^{14} + \alpha^{10} x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	2 decoder out	$c(x) = x^{14} + x^{13} + \alpha x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	3 decoder in	$c(x) = x^{14} + \alpha^2 x^{13} + \alpha^7 x^{12} + \alpha^4 x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + \alpha^4 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^8 x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
	3 decoder out	$c(x) = x^{14} + \alpha^2 x^{13} + \alpha^{14} x^{12} + 1 x^{11} + \alpha^2 x^{10} + \alpha^3 x^9 + 1 x^8 + \alpha^5 x^7 + \alpha^6 x^6 + \alpha^{10} x^5 + \alpha^9 x^4 + \alpha^{13} x^3 + x^2 + \alpha^{11} x + \alpha^{13}$
2	Data asal	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
	1 decoder in	$c(x) = \alpha^7 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$
	1 decoder out	$c(x) = \alpha^5 x^{14} + \alpha^6 x^{13} + \alpha^7 x^{12} + \alpha^7 x^{11} + \alpha^8 x^{10} + \alpha^9 x^9 + \alpha^{10} x^8 + \alpha^{11} x^7 + \alpha^{12} x^6 + \alpha^{13} x^5 + \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^{10} x^2 + \alpha^2 x + \alpha^3$

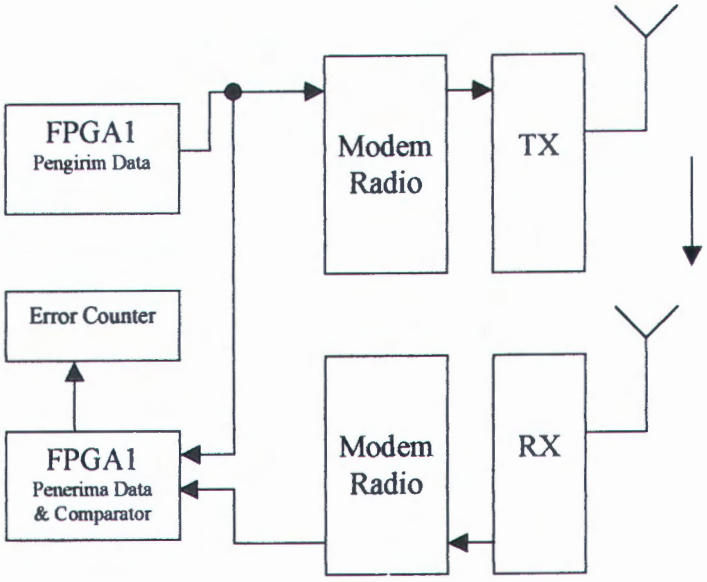
Tabel 4.6. Hasil pengujian decoder terhadap error burst (lanjutan)

Blok ke	Error Burst	Polinomial 3 codeword
	2 decoder in	$c(x) = \alpha^{13} x^{14} + \alpha^{9} x^{13} + \alpha^{7} x^{12} + \alpha^{7} x^{11} + \alpha^{8} x^{10} + \alpha^{9} x^{9} + \alpha^{10} x^{8} + \alpha^{11} x^{7} + \alpha^{12} x^{6} + \alpha^{13} x^{5} + \alpha^{14} x^{4} + \alpha^{3} x^{3} + \alpha^{10} x^{2} + \alpha^{2} x + \alpha^{3}$
	2 decoder out	$c(x) = \alpha^{5} x^{14} + \alpha^{6} x^{13} + \alpha^{7} x^{12} + \alpha^{7} x^{11} + \alpha^{8} x^{10} + \alpha^{9} x^{9} + \alpha^{10} x^{8} + \alpha^{11} x^{7} + \alpha^{12} x^{6} + \alpha^{13} x^{5} + \alpha^{14} x^{4} + \alpha^{3} x^{3} + \alpha^{10} x^{2} + \alpha^{2} x + \alpha^{3}$
	3 decoder in	$c(x) = \alpha^{5} x^{14} + \alpha^{14} x^{13} + 1 x^{11} + \alpha^{8} x^{10} + \alpha^{9} x^{9} + \alpha^{10} x^{8} + \alpha^{11} x^{7} + \alpha^{12} x^{6} + \alpha^{13} x^{5} + \alpha^{14} x^{4} + \alpha^{3} x^{3} + \alpha^{10} x^{2} + \alpha^{2} x + \alpha^{3}$
	3 decoder out	$c(x) = \alpha^{5} x^{14} + \alpha^{14} x^{13} + 1 x^{11} + \alpha^{8} x^{10} + \alpha^{9} x^{9} + \alpha^{10} x^{8} + \alpha^{11} x^{7} + \alpha^{12} x^{6} + \alpha^{13} x^{5} + \alpha^{2} x^{4} + \alpha^{4} x^{3} + \alpha^{10} x^{2} + \alpha^{2} x + \alpha^{3}$
3	Data asal	$c(x) = \alpha^{6} x^{8} + \alpha^{7} x^{7} + \alpha^{8} x^{6} + \alpha^{9} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$
	1 decoder in	$c(x) = \alpha^{13} x^{14} + \alpha^{6} x^{8} + \alpha^{7} x^{7} + \alpha^{8} x^{6} + \alpha^{9} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$
	1 decoder out	$c(x) = \alpha^{6} x^{8} + \alpha^{7} x^{7} + \alpha^{8} x^{6} + \alpha^{9} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$
	2 decoder in	$c(x) = \alpha^{7} x^{14} + \alpha^{5} x^{13} + \alpha^{6} x^{8} + \alpha^{7} x^{7} + \alpha^{8} x^{6} + \alpha^{9} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$
	2 decoder out	$c(x) = \alpha^{6} x^{8} + \alpha^{7} x^{7} + \alpha^{8} x^{6} + \alpha^{9} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$
	3 decoder in	$c(x) = \alpha^{8} x^{13} + \alpha^{7} x^{12} + 1 x^{11} + \alpha^{6} x^{8} + \alpha^{7} x^{7} + \alpha^{8} x^{6} + \alpha^{9} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$
	3 decoder out	$c(x) = \alpha^{4} x^{13} + \alpha^{13} x^{12} + \alpha^{10} x^{11} + \alpha^{11} x^{10} + \alpha^{3} x^{9} + \alpha^{5} x^{8} + \alpha^{6} x^{7} + \alpha^{7} x^{6} + \alpha^{4} x^{5} + \alpha^{10} x^{4} + \alpha^{5} x^{3} + \alpha^{13} x^{2} + \alpha^{8} x + \alpha^{10}$

Dari hasil di atas dapat dinyatakan bahwa: jika kanal yang dilewati mempunyai error burst sepanjang dari 3 hingga 6 pulsa clock, pengiriman simbol secara paralel dengan menggunakan modulasi -modulasi M-ary, akan mengakibatkan kerusakan sebanyak 3 hingga 6 simbol juga, Akibatnya codeword tidak dapat diperbaiki oleh decoder RS dengan t = 2 ini. Tetapi, dengan menggunakan pengiriman serial dengan modulasi – modulasi biner, error burst sepanjang 3 hingga 6 pulsa clock hanya mengakibatkan kerusakan maksimal dua simbol saja. Bagi decoder RS dengan t = 2 , kerusakan simbol sebanyak 2 buah. masih dapat diperbaiki. Maka pengiriman data serial modulasi M-ary yang menggunakan codec RS mempunyai keunggulan di banding pengiriman data paralel modulasi M-ary yang menggunakan codec RS, karena codec RS memproses sebuah simbol secara bersama-sama/paralel.

4.2. Evaluasi Codec Pada Kanal Radio

Untuk mengevaluasi *codec* pada komunikasi data lewat kanal radio, maka yang harus dilakukan adalah : menguji unjuk kerja komunikasi data lewat kanal radio **sebelum** dipasang codec RS dan **setelah** dipasang codec RS. Data informasi dikirim oleh ROM internal FPGA1, sedang penerimaan data dan komparasinya dilakukan di FPGA2. Diagram blok pengujian ditunjukkan dalam gambar 4.6.



Gambar 4.6. Diagram blok pengujian komunikasi data sebelum dipasang codec RS

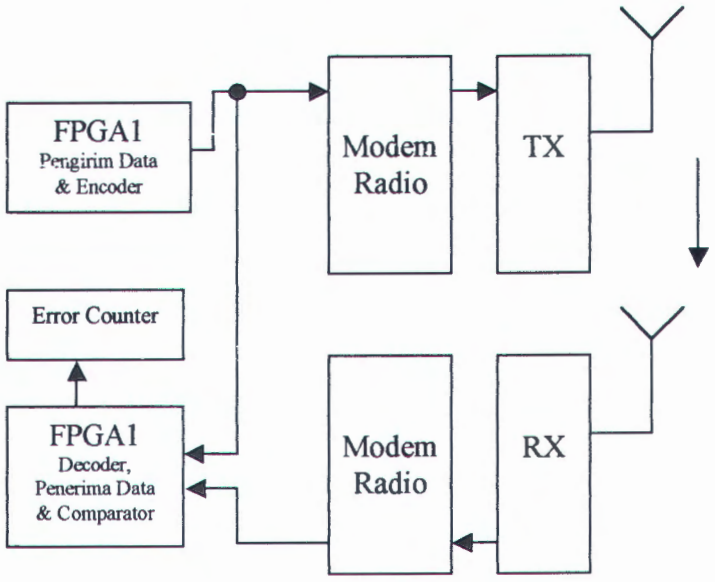
Dengan jumlah informasi sebanyak 10000 simbol, rangkaian diuji pada beberapa nilai level sinyal terima (dB μV). Data yang diterima lewat kanal radio dibandingkan langsung dengan data yang diterima lewat kabel pendek. Ketidaksamaan sebuah simbol antara kedua data identik dengan sebuah error. Jumlah error dihitung oleh error counter. Hasil pengujian untuk beberapa nilai sinyal terima, ditunjukkan dalam tabel 4.7.

Tabel 4.7. Hasil pengujian komunikasi data tanpa codec

Level Sinyal (dBμV)	40	35	30	26	22	20
Jumlah Simbol Error	0	0	1	0	6	4092

Dalam tabel 4.7 tampak bahwa error hampir tidak pernah terjadi pada level sinyal terima sebesar 26 hingga 40 dBμV. Error mulai terjadi pada level sinyal terima 22 dBμV dan terus bertambah seiring menurunnya kuat sinyal terima. Pada level sinyal 20 dBμV, jumlah error bertambah dengan cepat, karena sinyal sinkronisasi yang dikirim mengalami kerusakan. Pada level sinyal terima 20 dBμV, pengulangan pengujian akan sulit menghasilkan jumlah error, yang sama, karena pada saat itu sinyal sinkronisasi mengalami kerusakan, sehingga kerja rangkaian decoder tidak akan sama pula dari satu pengukuran ke pengukuran lain.

Diagram blok pengujian komunikasi data setelah di pasang codec ditunjukkan dalam gambar 4.7. Pada pengujian komunikasi setelah dipasang codec, pengirim data dan encoder di program pada FPGA1. Sementara itu, penerima data, comparator dan decoder, diprogram pada FPGA2.



Gambar 4.7. Diagram blok pengujian komunikasi data setelah dipasang codec RS

Pengujian dilakukan dengan data dan cara yang serupa dengan pengujian tanpa codec. Hasil pengujian ditunjukkan dalam tabel 4.8.

Tabel 4.8. Hasil pengujian komunikasi data dengan codec

Level Sinyal (dBμV)	40	35	30	26	22	20
Jumlah Simbol Error	0	0	0	0	0	3145

Dalam tabel 4.8 tampak bahwa error pada level sinyal terima sebesar 26 hingga 40 dBμV, dapat diperbaiki. Tetapi perbaikan tidak berjalan baik pada level sinyal terima 20 dBμV, karena sinyal sinkronisasi yang diterima mengalami kerusakan. Dengan kondisi ini, jumlah error yang diperbaiki tidak akan selalu sama jika dilakukan proses pengukuran ulang.

Hasil pada tabel 4.7 dapat dibandingkan pada tabel 4.8 dengan menggunakan format $P_{e\ simbol}$ fungsi SNR. Untuk itu jumlah simbol error pada tabel 4.7 dan 4.8 diubah ke format $P_{e\ simbol}$ dengan menggunakan persamaan :

$P_{e\ simbol}$ = jumlah simbol error/jumlah simbol data

Sedangkan pengubahan level sinyal terima (dalam dBμV) menjadi SNR (Signal to Noise Ratio) dilakukan dengan cara berikut ini.

Nilai daya N (Noise) pada Signal to Noise Ratio (SNR) didekati dengan menggunakan persamaan :

$N = K\ T\ B$

Di mana :

N = Noise

K = Konstanta Boltzman (1.38×10^{-23} Watts/K/Hz)

T = Temperatur (° Kelvin)

B = Bandwidth sistem (Hz)

Nilai – nilai yang digunakan adalah :

T = 273 + 27 ° C = 300 °K

B_r = Bandwidth FM stereo = 200 KHz = 200.000 Hz

Sehingga noisenya adalah :

$N = 1.38 \times 10^{-23} \times 300 \times 200.000 = 8{,}28 \times 10^{-16}$ W

Level sinyal terima (dBμV) merupakan perbandingan antara tegangan sinyal yang diterima (μV) dengan tegangan 1 μV dengan persamaan :

$$\text{Level sinyal terima (dB}\mu\text{V)} = 20 \log \frac{\text{tegangan sinyal terima (}\mu\text{V)}}{1\ (\mu\text{V})\}}$$

Dengan level sinyal terima yang sudah diketahui, tegangan sinyal terima dapat dihitung dengan persamaan :

Tegangan sinyal terima = $\log^{-1}$ (level sinyal/20) μV

Selanjutnya, daya S dapat dihitung dengan persamaan

$S = V^2 / Z$ Watt

Di mana :

V = tegangan sinyal terima (Volt)

Z = impedansi penerima (Ω)

Impedansi penerima yang digunakan pada evaluasi adalah sebesar 75 Ω. Dengan diperolehnya S dan N, maka SNR dapat dihitung dengan menggunakan persamaan :

$$SNR = 10 \log S/N \quad dB$$

Berdasar persamaan-persamaan tersebut, transformasi tabel 4.7 dan tabel 4.8 menjadi tabel $P_{e\ simbol}$ fungsi SNR ditunjukkan dalam tabel 4.9.

Tabel 4.9 Transformasi ke bentuk $P_{e\ simbol}$ fungsi SNR

Level sinyal (dBμV)	S (W)	SNR (dB)	Jumlah Simbol Error		$P_{e\ simbol}$	
			Non Codec	Dgn Codec	Non Codec	Dgn Codec
40	$1,3 \times 10^{-10}$	52	0	0	0	0
35	$4,2 \times 10^{-11}$	47	0	0	0	0
30	$1,3 \times 10^{-11}$	42	1	0	10^{-4}	0
26	$5,3 \times 10^{-12}$	38	0	0	0	0
22	$2,1 \times 10^{-12}$	34	6	0	6×10^{-4}	0
20	$1,3 \times 10^{-12}$	32	4092	3145	0,4	0,3

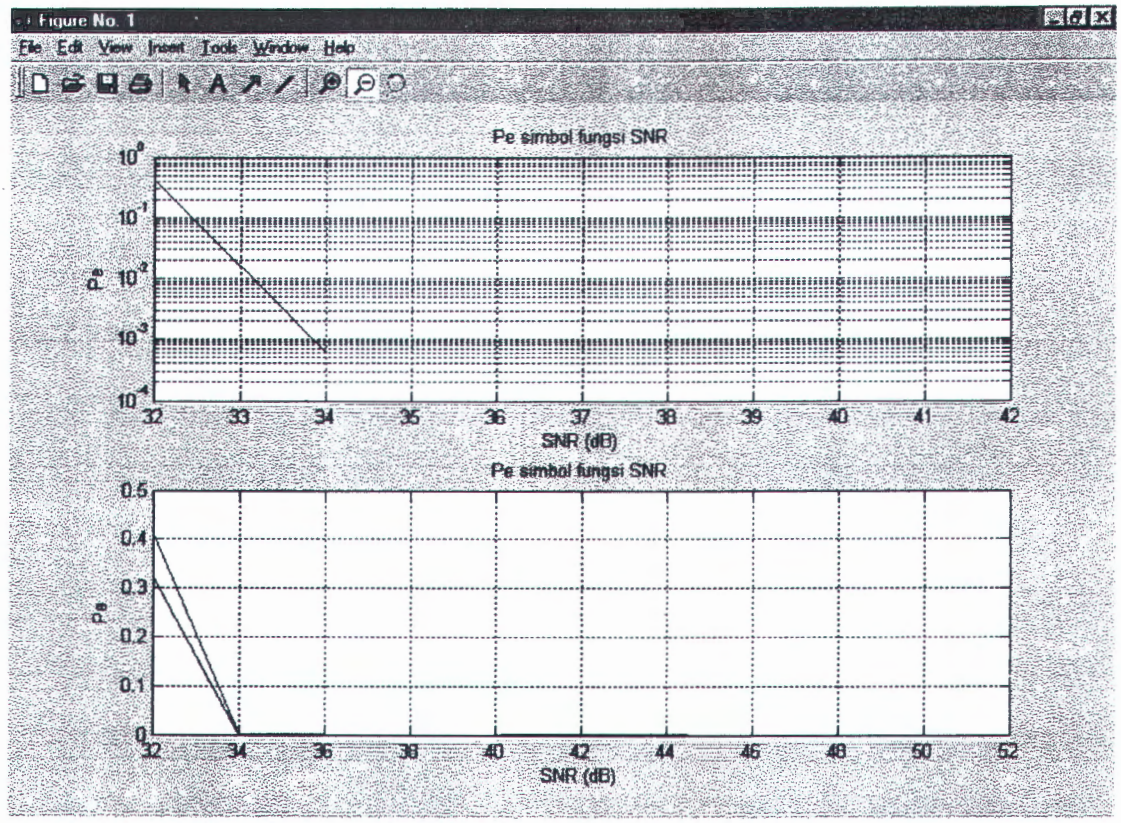
Tabel 4.9 menunjukkan bahwa error simbol yang terjadi pada SNR 42 dB dan 34dB ketika dikirim tanpa codec, dapat diperbaiki oleh codec sehingga pada kedua SNR tersebut $P_{e\ simbol}$ menjadi 0.

Jumlah error yang kecil antara 38 hingga 52 dB (pada level sinyal 26 hingga 40 dBμV, terjadi karena jumlah simbol yang digunakan sebagai masukan hanya 10.000. Sedangkan secara teoritis, untuk modulasi FSK P_e adalah 10^{-4}. Artinya di antara 10.000 data, kemungkinan hanya ada satu error yang terjadi. Penambahan data sulit dilakukan karena keterbatasan peralatan percobaan. Untuk itu perlu metode/ rangkaian lain untuk menghasilkan data yang lebih banyak.

Peningkatan error yang tinggi pada SNR 32 dB terjadi karena pada titik ini, sinyal sinkronisasi mengalami kerusakan stabilitas. Akibatnya penerima tidak dapat mengidentifikasi data secara benar. Penggunaan codec pada kondisi ini juga tidak banyak membantu, karena codec juga memerlukan sinyal sinkronisasi. Level sinyal terima pada SNR 32 dB ini adalah 20 dBμV. Nilai ini jauh lebih kecil daripada standard level sinyal minimal untuk FM Stereo Broadcast yang sebesar 50 dBμV. Standar ini dijadikan bahan perbandingan karena pemancar dan penerima untuk evaluasi menggunakan pemancar dan penerima FM stereo pada frekuensi 100 MHz. Masih memungkinkannya perolehan $P_{e\ simbol}$ kecil di bawah level standar minimum 50 dBμV, karena informasi yang dikirim adalah informasi digital (bukan analog).

Pada SNR 34 dB, nilai $P_{e\ simbol}$ yang sedikit lebih banyak daripada 1, terjadi karena daerah ini merupakan transisi menuju ke daerah level minimum. Pada daerah ini, codec masih dapat bekerja, karena gangguan terhadap sinyal sinkronisasi relatif kecil. Semakin dekat ke 32 dB, jumlah error akan semakin meningkat. Dari tabel 4.9 dapat dibuat grafik $P_{e\ simbol}$ fungsi SNR seperti dalam gambar 4.8.

Pada gambar 4.8, dengan skala P_e logaritmis, perbedaan antara P_e tanpa codec dan dengan codec tidak tampak. Penggunaan skala logaritmik menyebabkan perbedaan antara keduanya sangat kecil. Bentuk grafik yang lurus antara 32 hingga 34 dB terjadi karena pengambilan titik pengukuran di antara kedua nilai tersebut tidak dapat dilakukan. Penyebabnya adalah ketidakakuratan skala pada Synthesized FM-TV Field Strength Meter serta kesulitan setting alat untuk mendapat titik-titik pengujian di antara dua nilai tersebut.



Gambar 4.8. Grafik $P_{e\ simbol}$ fungsi SNR

Pada skala linier, grafik di atas merupakan grafik tanpa codec, sedang grafik dibawahnya merupakan grafik dengan codec. Dari grafik tersebut, dapat dinyatakan bahwa setelah dipasang codec, terjadi penurunan P_e simbol untuk daerah antara 32 hingga 34 dB. Maka adanya codec telah meningkatkan unjuk kerja dari sistem komunikasi.

4.3. Pembahasan

Dari tahap perencanaan hingga tahap pengukuran/pengujian, cukup banyak kendala yang ditemui. Salah satu yang penting adalah sinyal sinkronisasi pengukuran

yang tidak sama dengan sinyal sinkronisasi rencana. Akibatnya rangkaian codec tidak dapat bekerja dengan baik. Kekurangan ini terjadi karena sulitnya mendapat informasi karakteristik waktu dari komponen dan rangkaian. Kalaupun ada, tidak menyentuh hal yang diinginkan.

Selain itu, spesifikasi tiap elemen rangkaian dalam FPGA (seperti fan in, fan out, dan lainnya) tidak diperoleh. Akibatnya, rangkaian menjadi tidak bekerja ketika dibebani. Dari pengalaman selama merencanakan hingga pengujian, ditemui banyak masalah dan penyelesaian yang tidak sama. Sehingga sulit untuk digeneralisasi.

Dari segi kecepatan data, FPGA tampaknya cukup fleksibel. Hal ini terbukti, karena di beberapa pengujian yang memerlukan kecepatan data berbeda (0,5 bps hingga 500 bps), encoder dan decoder masih bekerja dengan baik. Sebenarnya FPGA mampu bekerja pada kecepatan jauh di atas 500 bps, tetapi terbatasnya kemampuan modem yang hanya sebesar 1200 bps membuat pengujian hanya dilakukan pada kecepatan tersebut.

Untuk penerapan di lapangan, codec yang telah dirancang perlu diuji pada beberapa kondisi pembebanan, dan kecepatan pengiriman data. Selain itu, perbaikan perlu dilakukan pada bagian-bagian yang kerjanya belum konsisten, jika dipasang beban yang berbeda. Semakin sedikit bagian rangkaian yang tidak konsisten, kinerja codec akan semakin meningkat.

BAB V
PENUTUP

5.1. Kesimpulan

Setelah melakukan proses rancang bangun CODEC RS (15,11,2) mulai dari mencari referensi hingga melakukan pengujian, penyusun mendapatkan beberapa kesimpulan sebagai berikut :

- Bagian-bagian rangkaian codec RS(15,11,2) telah bekerja sesuai dengan yang diinginkan.
- Codec bekerja sesuai dengan kemampuan koreksi error maksimal sebesar 2 simbol.
- Karena data dikirim secara serial, maka codec RS (15,11,2) mampu melakukan perbaikan terhadap codeword yang terganggu burst, selama error hanya terjadi paling banyak pada 2 simbol
- Saat dievaluasi pada kanal radio, codec RS (15,11,2) mampu melakukan koreksi dengan baik, ketika sinyal sinkronisasi tidak rusak (pada SNR 54 dB hingga 32 dB).
- Sinkronisasi menentukan kesuksesan proses encoding dan decoding.
- Kesuksesan proses rancang bangun juga ditentukan oleh data akurat karakteristik diagram waktu dari perangkat keras Codec dan perangkat keras sistem komunikasi digitalnya, karena sistem bekerja secara real time.

5.2. Saran

Untuk mendapatkan hasil rancang bangun yang lebih baik, penyusun mempunyai saran-saran sebagai berikut

- Karakteristik diagram waktu dari Codec dan sistem komunikasi digital harus dirancang dengan sangat cermat, karena delay-delay yang tidak diperhitungkan akan dapat merusak kerja sistem.
- Tiap delay , sekecil apapun yang dihasilkan oleh bagian perangkat keras, harus segera diatasi agar tidak menjalar ke rangkaian-rangkaian berikutnya.
- Perlu dipikirkan cara pengiriman sinyal sinkronisasi yang lebih cermat dan praktis.

DAFTAR PUSTAKA

[1] Alfke, P. (1998) *Xilink FPGAs : A Technical Overview for the First-Time User*, XAPP.

[2] Anderson, J.B., Mohan, S. (1991) *Source and Channel Coding, An Algorithmic Approach*, Kluwer Academic Pubh., Massachusetts.

[3] Breeding, K.J. (1989) *Digital Design Fundamentals, Second Edition.* Prentice Hall, Inc. , New Jersey.

[4] Chio, A.A.P., Shagum, J.A., Sabido IX, D.J.M. (--) *VLSI Implementation of (255,223) Reed-Solomon Error Correction Codec. Proceeding of 2ND National ECE Conference (Philipines).*

[5] Fanebust, H. dan Gronstad, S. dan Rislow, B. dan Rognerud, T. (--) *Reconfigurable Multimedia Modem for Satellite Communication.* Nera Research, Norway.

[6] Gill, J. (2002), EE387 Lecture, Stanford Univerity, Stanford.

[7] Lee, H. dan Yu, M.L. dan Song L. (--) *VLSI Design of Reed-Solomon Decoder Aschitectures.* University of Minnesota, Minneapolis.

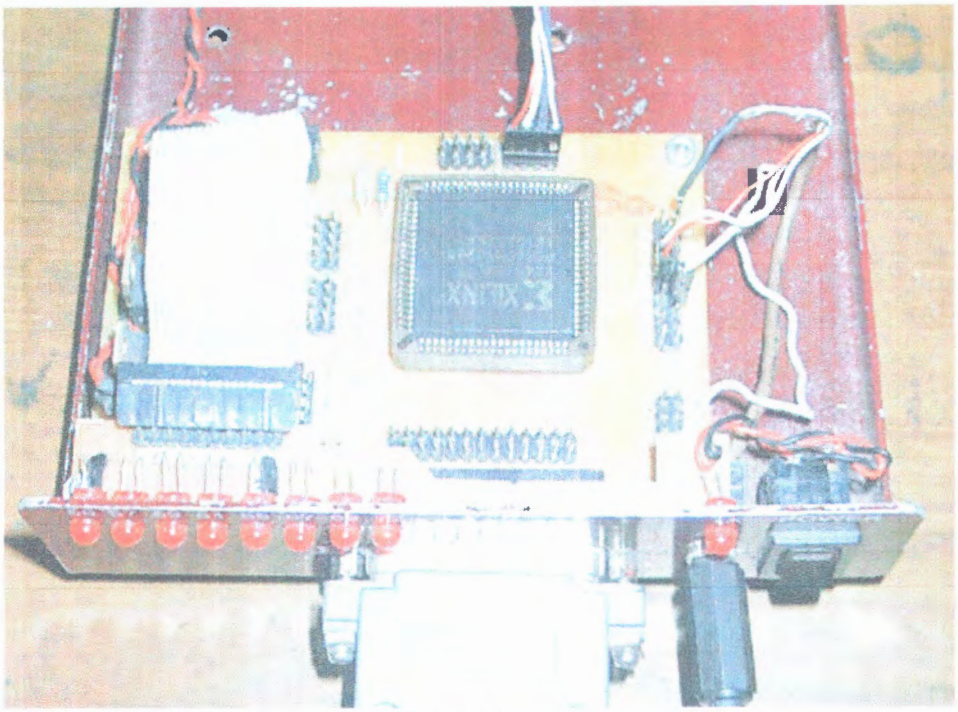
[8] Lin, S., dan Costello Jr., D. J. (1983) *Error Control Coding, Fundamentals and Application.* Prentice Hall, Inc., New Jersey.

[9] Lomena, A.G. dan Lopez, J.C. dan Royo, A. (--) *A Pipeline Frequency Domain Reed-Solomon Decoder for Application in ATM Neyworks.* Universidad Polytechnica de Madrid, Madrid.

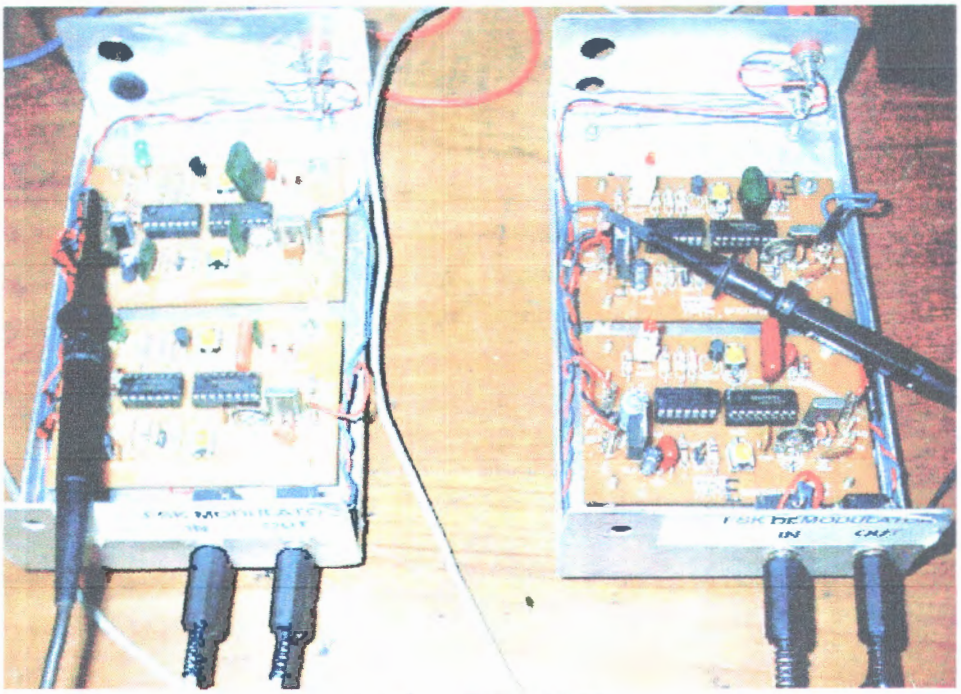
[10] Michelson, A.M. dan Levesque, A. H. (1985) *Error Control Techniques for Digital Communication.* John Willey & Sons , New York.

[11] Rappaport, T.S. (1996) *Wireless Communication, Principles & Practice.* Prentice Hall, Inc., New Jersey.

[12] Shanmugam, K.S. (1979) *Digital and Analog Communication System.* John Willey & Sons , New York.



FPGA XC4010XL



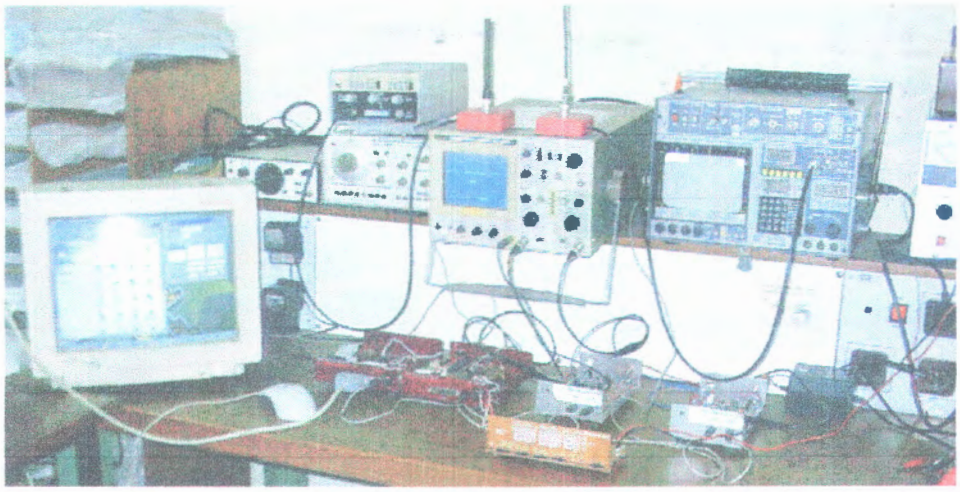
Modem TCM-3105

Peralatan yang digunakan

Stereo Coder Koenig SC-600B

Pengukur Kuat Sinyal Antena Koenig APM 741

Oscilloscope National VP 5220A



LAMPIRAN

1. Spesifikasi peralatan
   Stereo Coder Koenig SC-600B

TECHNISCHE DATEN

MULTIPLEX-AUSGANG

Betriebsart:	L = R, L = -R, L und R (Pilotton kann zugeschaltet werden)
Ausgangsspannung:	5 V_{ss} ± 10 % stufenlos regelbar
Pilotfrequenz:	19 kHz ± 2 Hz (quarzstabilisiert)
Hilfsträgerfrequenz:	38 kHz ± 4 Hz
Phasendifferenz zwischen Hilfsträger und Pilot:	≦ 3°
Ausgangsimpedanz:	≧ 2,5 kΩ
Innenwiderstand:	200 Ω
Hilfsträgerunterdrückung:	≧ 40 dB
Übersprechdämpfung linker Kanal zu rechtem Kanal:	intern ≧ 40 dB extern ≧ 40 dB (bis 10 kHz)

MODULATIONSARTEN

Intern:	80 - 400 - 1000 - 5000 - 10 000 Hz ± 5 % wahlweise schaltbar
NF-Klirrfaktor:	≦ 0,6 %
Extern:	5-poliger DIN-Eingang
Frequenzbereich:	30 Hz - 15 kHz
Eingangswiderstand:	500 kΩ
Preemphasis:	50 µs ± 1 dB
max. Eingangsspannung:	10 V_{ss}

HF-AUSGANG

Ausgangsspannung:	800 µV_{eff}
Ausgangsimpedanz:	75 Ω
Frequenz:	100 MHz ± 1 MHz

INTERNE MODULATION

Frequenzhub:	Multiplex 40 kHz
	Pilotton 6,75 kHz

EXTERNE MODULATION

Hub:	0 - 75 kHz (100 %)
Modulationskoeffizient:	10 mV_{eff} / kHz (f ≦ 1 kHz)

Der Modulationskoeffizient am Multiplexausgang entspricht 1 V_{ss} bei 8 kHz Hub.

NF-SYNCHRONISIERUNGSAUSGANG

Ausgangspegel:	3 V_{ss}
Innenwiderstand:	10 kΩ
Frequenz:	80 Hz, 400 Hz, 1 kHz, 5 kHz, 10 kHz
Klirrfaktor:	≦ 0,1 % 80 Hz - 5 kHz
	≦ 0,25 % bei 10 kHz
	typisch 0,02 % bei 1 kHz

VERSORGUNG

Spannung:	127 oder 220 V umschaltbar
Frequenz:	50 Hz
Leistungsaufnahme:	3 Watt
Abmessungen:	103 x 165 x 260 mm (H x B x T)
Gewicht:	1,3 kg
Zubehör:	Bedienungsanleitung
	Netzkabel
	Verbindungskabel - BNC-Bananenstecker
	1 Sicherung 0,125 A

TECHNISCHE DATEN

EINGÄNGE

Empfangsbereiche:	VHF Band I (Kanal 2 - 4)	47 - 68 MHz
	Unteres Sonderkanalband (Kanal S 1 - S 10)	105 - 174 MHz
	VHF Band III (Kanal 5 - 12)	174 - 230 MHz
	Oberes Sonderkanalband (Kanal S 11 - S 20)	230 - 300 MHz
	Hyperband (Kanal S 20 - S 41)	302 - 470 MHz
	UHF Band IV / V (Kanal 21 - 69)	470 - 860 MHz
	ZF Band (BT)	38,9 MHz
	UKW/FM-Band	88 - 108 MHz
	Satellitenband (mit Option SR 815)	950 - 1750 MHz
Frequenzanzeige:	4 stellige Digitalanzeige (LCD) Auflösung 100 kHz für TV;	
	4 stellige Digitalanzeige (LCD) Auflösung 1 MHz bei SAT (mit Option SR 815);	
	5 stellige Digitalanzeige (LCD) Auflösung 10 kHz für UKW/FM; keine Anzeige für ZF	
Frequenzgenauigkeit:	Anzeigewert x 10 ± 1 Digit	
Kanaleingabe:	Synthesizer-Abstimmung mit 99 Kanälen programmiert auf die CCIR Kanäle für die Normen B, G und H.	
Feinverstimmung:	in 50 kHz-Schritten; max. ± 4 MHz	
Kanalanzeige:	2 stellige LCD-Anzeige	
Kanalspeicher:	30 Speicherplätze für TV-Kanäle	

50

Pegelmeßbereich:	20 dBµV - 130 dBµV (ZF 60 - 130 dBµV)
Impedanz:	75 Ω unsymmetrisch
Eingangsschutz:	max. 100 V DC
Genauigkeit:	› ± 3 dB in Band I, III, ZF u. UKW/FM › ± 6 dB in Band IV/V und Hyperband (unter Berücksichtigung der Korrekturkurve)
Videoeingang:	1 V_{ss} /75 Ω
Voltmetereingang:	5 - 50 V (DC oder AC) ± 5 %

AUSGÄNGE

Videoausgang:	1 V_{ss} /75 Ω
Tonausgangsleistung:	max. 0,3 Watt (über Lautsprecher)
UKW/FM-Stereoausgang:	Stereo-Kopfhöreranschluß; Impedanz › 8 Ω (nur für UKW/FM)
Gleichspannung:	ca. 12 V / max. 50 mA

ALLGEMEINES

Stromversorgung:	220 V~ ± 10 % / 50 Hz oder 12 V DC (aus nachrüstbarem Akku) oder 12 V - 18 V DC/2 A externe Quelle (z.B. Autobatterie)
Leistungsaufnahme:	max. 46 W (ohne Text) max. 53 W (mit Text)
Empfohlener Akku:	12 V / 5,7 Ah
Bildröhre:	6 Zoll (15 cm)
Arbeitstemperaturbereich:	0 - 40° C
Abmessungen:	340 x 180 x 440 mm (B x H x T) (mit aufgesetztem Schutzdeckel)
Gewicht:	8,5 kg (ohne Schutzdeckel)
Zubehör:	Netzkabel Adapter BNC-Koax-Buchse (P 80) Adapter BNC-Koax-Stecker (P 81) Tragegurt

Lichtschutztubus
Ersatzsicherungen (3,15 A)
Stecker für ext. 12-Volteingang
2 Meßkabel für Voltmeter
Umrechnungsschieber dB/dBµV
Vergleichstabelle Frequenz/Kanal (im Deckel)
Bedienungsanleitung

Außer der Bedienungsanleitung ist sämtliches Zubehör im Schutzdeckel untergebracht.

Zusätzlich lieferbares Zubehör:

stabiler Transportkoffer
(Best.-Nr. 5585)

Satellitenreceiver SR 815
(Best.-Nr. 5658)

Satellitenkonverter SAC 870
(Best.-Nr. 5622)

Rauschgenerator NG 75 + P 136
(Best.-Nr. 5660)

Videotextmodul für APM 742
(Best.-Nr. 5683)

FUNKTIONEN

Monitor:	Normales komplettes Fernsehbild
Zoom:	Bild wird in horizontaler Richtung um das Zweifache vergrößert (nur noch Bildausschnitt sichtbar)
Pegel:	Weißer Leuchtbalken oben im Bildschirm, der den Analogwert des Pegels anzeigt.
Spektrum: (breitbandig)	Darstellung aller Sender, die in einem Bandbereich empfangen werden können.
Spektrum: (schmalbandig)	Darstellung von Bild- und Tonträger eines Kanals oder mehrerer Kanäle.
Austastlücke:	Fernsehbild nach rechts verschoben - links ist die H-Austastlücke mit dem Oszillogramm des H-Austastimpulses und des Bursts sichtbar.
Voltmeter:	Weißer Leuchtbalken oben im Bild, wie bei "Pegel"; es wird lediglich auf einer zweiten Skala abgelesen.
Akustisches Signal:	Meßton, der sich je nach Eingangspegel in der Tonhöhe verändert.
Stereo:	Ein "S" im Kanaldisplay erscheint beim Empfang eines Stereosenders.
Ton:	Normaler Ton des jeweils eingestellten Fernseh-, Satelliten- oder UKW-Senders.
Videotext: (APM 742 TXT)	Darstellung aller Videotexttafeln auf dem Bildschirm.