

TEKNIK CRC DALAM ERROR CONTROL CODING

Ratnasari Nurrahmah

Jurusan Teknik Elektro Fakultas Teknik UMS

Jl. A.Yani Pabelan-Kartasura, Tromol Pos 1 Surakarta

ABSTRAK

Kerusakan data merupakan masalah yang penting dalam peng-irisan maupun penyimpanan data. Teknik pendeteksian error dengan error control coding akan sangat membantu dalam mempertahankan keaslian data. Hampir semua metode pendeteksian error dilakukan dengan pengkodean data yang menambahkan bit ekstra pada bit-bit data yang akan ditransmisikan. Salah satu metode error control coding adalah Cyclic Redundancy Check atau CRC yang dapat diimplementasikan secara sederhana baik secara hardware maupun software. Makalah ini memperlihatkan komputasi matematis dari Cyclic Redundancy Check dan implementasi hardware-nya.

Kata kunci : Pendeteksian error, error control coding, Cyclic Redundancy Check

PENDAHULUAN

Kerusakan data merupakan masalah penting dalam transmisi dan penyimpanan data. Kerusakan data yang ditransmisikan melalui suatu media komunikasi yang berupa timbulnya error atau kesalahan, disebabkan oleh berbagai faktor seperti adanya noise dalam kanal komunikasi, *fading* dari sinyal, interferensi antar kanal dan sebagainya. Suatu penerima harus dapat membedakan antara data yang mengandung error dan yang bebas dari error. Teknik yang digunakan untuk mengurangi terjadinya error ini disebut dengan *Error Control Coding*, yaitu pengkodean data dengan orientasi pendeteksian error dengan cara menambahkan bit-bit tambahan pada message bit.

Berbagai metode digunakan dalam pendeteksian error, diantaranya adalah

dengan *parity check codes* (baik vertikal maupun horisontal), *longitudinal redundancy check* (LRC), *checksums*, dan *cyclic redundancy check* (CRC). Adapun yang sekarang banyak digunakan adalah teknik CRC. Implementasi CRC ini dapat dilakukan secara *hardware* maupun *software*. Implementasi hardware yang dilakukan secara sederhana dengan menggunakan suatu rangkaian register geser, data ditangani per satu bit untuk satu waktu. Dalam implementasi software, data diproses untuk tiap byte atau bahkan tiap word dalam satu waktu sehingga proses akan berlangsung lebih cepat.

CYCLIC REDUNDANCY CODES

Metode ini dilakukan dengan cara menambahkan bit-bit tertentu dalam suatu frame message dengan suatu frame yang

disebut dengan *frame check sequence* (FCS) sebelum ditransmisikan oleh pemancar. FCS ini adalah sedemikian rupa sehingga frame yang ditransmisikan dapat dibagi secara eksak (tanpa sisa) dengan pembagi berpola tertentu. Pendeteksian error data dilakukan oleh penerima dengan cara membagi frame yang diterima dengan pembagi berpola sama dengan pola pembagi pada pemancar, jika dalam pembagian ini tidak terdapat sisa (hasilnya eksak), maka dapat disimpulkan bahwa tidak terjadi error dalam pengiriman frame tersebut. Jika panjang dari message adalah k bit dan panjang dari FSC adalah n bit, maka panjang frame yang akan ditransmisikan adalah $n + k$ bit.

Untuk lebih jelasnya, jika kita misalkan:

M = message dengan panjang k -bit

F = FCS dengan panjang n -bit

T = frame yang akan ditransmisikan dengan panjang $(k + n)$ -bit, dengan n bit terakhir adalah FCS

P = angka pembagi tertentu (pola) dengan panjang $n + 1$ bit.

Maka frame yang akan ditransmisikan adalah:

$$T = 2^n M + F \quad (1)$$

Perkalian M dengan 2^n diatas akan menyebabkan efek penggeseran message (M) n -bit ke kiri dan menutup hasilnya (*padded out*) dengan angka 0, sehingga didapatkan data dengan panjang $k + n$ bit dimana n bit terakhir adalah deretan angka 0. Hasil penggeseran ini kemudian akan ditambah dengan n -bit FCS untuk mendapatkan frame yang akan ditransmisikan (T). Dengan cara ini maka frame yang akan ditransmisikan (T) adalah sederetan $k + n$ bit, dimana k bit yang pertama adalah bit-bit dari message dan n bit yang terakhir adalah FCS.

Jika pembagian T/P diinginkan untuk tidak terdapat sisa maka kita harus

menentukan FCS sedemikian rupa sehingga:

$$\frac{T}{P} = \frac{2^n M + F}{P} = Q \quad (2)$$

dimana Q adalah hasil bagi eksak.

Misalkan jika kita bagi $2^n M$ dengan P kita dapatkan hasil bagi dan suatu sisa, yakni:

$$\frac{2^n M}{P} = Q + \frac{R}{P} \quad (3)$$

dimana Q adalah hasil bagi dan R adalah sisa hasil bagi.

Untuk mendapatkan hasil bagi T/P yang eksak, misalkan kita gunakan R sebagai FCS, maka dengan menggunakan (1) kita dapatkan:

$$\frac{T}{P} = \frac{2^n M + R}{P} = \frac{2^n M}{P} + \frac{R}{P} = Q + \frac{R}{P} + \frac{R}{P} \quad (4)$$

Karena penambahan disini adalah merupakan penambahan modulo 2 (EXOR), maka $R/P + R/P$ akan sama dengan nol, dan:

$$\frac{T}{P} = Q$$

yang berarti mempunyai hasil bagi yang eksak.

Suatu contoh sederhana berikut ini akan menjelaskan prosedur di atas:

Suatu message $M = 1010001101$ (10 bit) akan ditransmisikan dengan pola pembagi $P = 110101$ (6 bit). Untuk mendapatkan frame yang akan ditransmisikan (T), terlebih dahulu kita tentukan FCS. Jika kita tuliskan M dan P serta $2^n M$ dalam bentuk polinomial, maka:

$$M(X) = X^9 + X^7 + X^3 + X^2 + X^1$$

$$P(X) = X^5 + X^4 + X^2 + 1$$

$$2^n M(X) = X^n M(X)$$

Karena pola pembagi mempunyai panjang 6 bit, maka panjang FCS adalah $n = 6 - 1 = 5$ bit. Dan karenanya :

$$2^n M(X) = X^5 M(X) = X^{14} + X^{12} + X^8 + X^7 + X^5$$

$$\frac{2^n M(X)}{P(X)} = \frac{X^{14} + X^{12} + X^8 + X^7 + X^5}{X^5 + X^4 + X^2 + 1}$$

Yang sama dengan 101000110100000. Untuk mendapatkan FCS data ini kita bagi dengan pola pembagi dan sisa hasil bagi merupakan FCS yang dimaksud:

Yang dapat dicari dengan perhitungan tangan:

$$\begin{array}{r} X^9 + X^8 + X^6 + X^4 + X^2 + X \\ \hline X^5 + X^4 + X^2 + 1 \overline{) X^{14} + X^{12} + X^8 + X^7 + X^5} \\ \underline{X^{14} + X^{13} + X^{11} + X^7} \\ X^{13} + X^{12} + X^{11} + X^9 + X^8 + X^7 + X^5 \\ \underline{X^{13} + X^{12} + X^{10} + X^8} \\ X^{11} + X^{10} + X^9 + X^7 + X^5 \\ \underline{X^{11} + X^{10} + X^8 + X^6} \\ X^9 + X^8 + X^7 + X^6 + X^5 \\ \underline{X^9 + X^8 + X^6 + X^4} \\ X^7 + X^5 + X^4 \\ \underline{X^7 + X^6 + X^4 + X^2} \\ X^6 + X^5 + X^2 \\ \underline{X^6 + X^5 + X^3 + X} \\ X^3 + X^2 + X \end{array}$$

Dari perhitungan tersebut didapatkan sisa hasil bagi dalam polinomial $X^3 + X^2 + X$ yang sama dengan 1110. Deretan bit ini yang akan kita gunakan sebagai FCS, karena FCS yang dikehendaki adalah sepanjang lima bit maka FCS = 01110. Dan frame transmisi T akan sama dengan:

$$T(X) = X^{14} + X^{12} + X^8 + X^7 + X^5 + (X^3 + X^2 + X)$$

$$= X^{14} + X^{12} + X^8 + X^7 + X^5 + X^3 + X^2 + X$$

Atau $T = 101000110101110$ dimana 10 bit yang pertama merupakan deretan bit dari message dan 5 bit terakhir merupakan deretan bit dari FCS. Deretan bit inilah yang akan ditransmisikan melalui saluran komunikasi menuju ke penerima.

Pada penerima pendeteksian error dilakukan dengan membagi data yang diterima dengan suatu pembagi yang berpola sama dengan pemancar. Pada kasus di atas jika data yang diterima oleh pemancar adalah $G(X) = X^{14} + X^{12} + X^8 + X^7 + X^5 + X^3 + X^2 + X$, maka data tersebut akan dibagi dengan $P(X) = X^5 + X^4 + X^2 + 1$, untuk keperluan

an pendeteksian error dapat dilihat pada rumus pendeteksian error di bawah.

Dari perhitungan di atas diperlihatkan bahwa tidak terdapat sisa dalam pembagian di atas, hal ini menandakan bahwa pengiriman data bebas dari kesalahan (*error free*).

PENGGUNAAN SHIFT REGISTER UNTUK ENCODING CRC

Proses CRC dapat dengan mudah diimplementasikan dengan menggunakan rangkaian berjalur umpan balik yang terdiri dari shift register dan gerbang EXOR. Rangkaian tersebut adalah sebagaimana langkah berikut:

Rumus pendeteksian error :

$$\begin{array}{r}
 X^9 + X^8 + X^6 + X^4 + X^2 + X \\
 X^6 + X^4 + X^2 + 1 \sqrt{\begin{array}{r} X^{14} + X^{12} + X^8 + X^7 + X^5 + X^3 + X^2 + X \\ X^{14} + X^{13} + X^{11} + X^7 \end{array}} \\
 \hline
 \begin{array}{r} X^{13} + X^{12} + X^{11} + X^9 + X^8 + X^7 + X^5 + X^3 + X^2 + X \\ X^{13} + X^{12} + X^{10} + X^8 \end{array} \\
 \hline
 \begin{array}{r} X^{11} + X^{10} + X^9 + X^7 + X^5 + X^3 + X^2 + X \\ X^{11} + X^{10} + X^8 + X^6 \end{array} \\
 \hline
 \begin{array}{r} X^9 + X^8 + X^7 + X^6 + X^5 + X^3 + X^2 + X \\ X^9 + X^8 + X^6 + X^4 \end{array} \\
 \hline
 \begin{array}{r} X^7 + X^5 + X^4 + X^3 + X^2 + X \\ X^7 + X^6 + X^4 + X^2 \end{array} \\
 \hline
 \begin{array}{r} X^6 + X^5 + X^3 + X \\ X^6 + X^5 + X^3 + X \end{array} \\
 \hline
 0
 \end{array}$$

1. Register terdiri dari n bit yang merupakan panjang dari FCS.
2. Gerbang EXOR maksimal adalah sejumlah n .
3. Ada atau tidaknya gerbang EXOR tergantung pada ada dan tidaknya *term* dalam polinomial pembagi, $P(X)$.

Menggunakan contoh di atas dimana:

Message $M(X) = X^9 + X^7 + X^3 + X^2 + X^1$

Pembagi $P(X) = X^5 + X^4 + X^2 + 1$

Maka register yang digunakan adalah register 5 bit dengan 3 gerbang EXOR. Implementasi untuk contoh di atas dapat dilihat pada Gambar 1 dan Tabel 1.

Proses dimulai dengan terlebih dahulu mereset register (*clear*). Message dimasukkan dalam register per satu bit, dimulai dari *Most Significant Bit*. Karena *feedback* hanya muncul setelah satu bit memasuki ujung register, maka sampai pada langkah ke empat (sampai pada bit keempat dari message) proses yang terjadi hanyalah proses pergeseran biasa. Saat satu

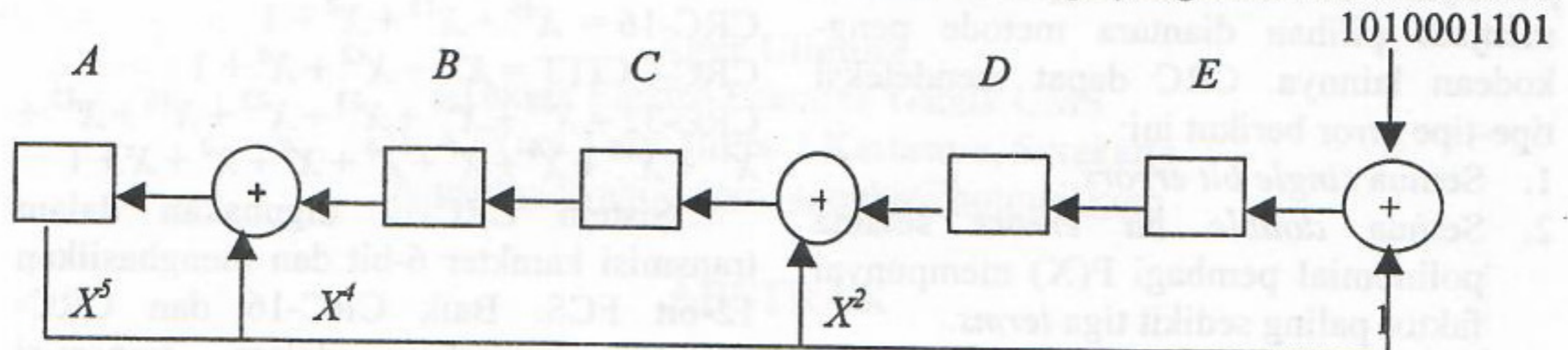
bit memasuki ujung register (langkah kelima), maka dengan adanya jalur *feedback*, bit ini akan di tambahkan (EXOR) pada pergeseran berikutnya (langkah keenam).

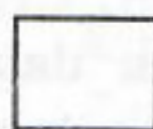
Tabel 1. memperlihatkan isi register untuk tiap selesainya proses satu langkah. Pada langkah ke-6, isi register adalah A=1, B=1, C=1, D=0, E=1 dan bukannya A=0, B=1, C=0, D=0, E=0 (pergeseran biasa tanpa adanya *feedback*). Jalur *feedback* yang ada menyebabkan ditambahkannya (EXOR) isi register A dari proses langkah kelima ("1") dengan bit masukan sebelum bit ini masuk register E, isi dari register D (hasil langkah ke-5) sebelum digeser ke C, dan isi dari register B (langkah ke-5) sebelum di geser ke A.


Proses tersebut berlangsung untuk semua bit-bit dalam message dan lima bit "0". Penambahan lima bit (00000) tambahan ini dilakukan untuk memperhitungkan pergeseran message M lima bit untuk

mengakomodasi FCS. Setelah bit terakhir diproses, register akan berisi sisa hasil bagi (FCS) yang akan ditransmisikan bersama-sama dengan bit-bit message.

Bit-bit message yang akan ditransmisikan



 : 1 bit shift register

 : EXOR

Gambar 1 Rangkaian dengan register geser untuk pembagian dengan polinomial $X^5 + X^4 + X^2 + 1$

Tabel 1. Rangkaian dengan register geser untuk pembagian dengan polinomial $X^5 + X^4 + X^2 + 1$

Register	A	B	C	D	E	Bit masukan
Isi awal	0	0	0	0	0	
Langkah 1	0	0	0	0	1	1
Langkah 2	0	0	0	1	0	0
Langkah 3	0	0	1	0	1	1
Langkah 4	0	1	0	1	0	0
Langkah 5	1	0	1	0	0	0
Langkah 6	1	1	1	0	1	0
Langkah 7	0	1	1	1	0	1
Langkah 8	1	1	1	0	1	1
Langkah 9	0	1	1	1	1	0
Langkah 10	1	1	1	1	1	1
Langkah 11	0	1	0	1	1	0
Langkah 12	1	0	1	1	0	0
Langkah 13	1	1	0	0	1	0
Langkah 14	0	0	1	1	1	0
Langkah 15	0	1	1	1	0	0

PENUTUP

Cyclic redundancy check yang cukup *powerfull* dan mudah diimplementasikan menjadi pilihan diantara metode pengkodean lainnya. CRC dapat mendeteksi tipe-tipe error berikut ini:

1. Semua *single bit errors*
2. Semua *double bit errors* selama polinomial pembagi $P(X)$ mempunyai faktor paling sedikit tiga *terms*.
3. Error dengan jumlah bit ganjil, selama polinomial pembagi $P(X)$ mempunyai faktor $(X + 1)$.
4. *Burst error* dengan panjang *burst* kurang dari panjang FCS.
5. *Burst error* yang lebih besar.

Polinomial pembagi dalam CRC sering juga diistilahkan sebagai generator

polinomial. Beberapa generator polinomial yang sering digunakan adalah:

$$\text{CRC-12} = X^{12} + X^{11} + X^3 + X^2 + X + 1$$

$$\text{CRC-16} = X^{16} + X^{15} + X^2 + 1$$

$$\text{CRC-CCITT} = X^{16} + X^{12} + X^5 + 1$$

$$\text{CRC-32} = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Sistem CRC-12 digunakan dalam transmisi karakter 6-bit dan menghasilkan 12-bit FCS. Baik CRC-16 dan CRC-CCITT digunakan dalam transmisi karakter 8-bit di USA dan EROPA. CRC-32 merupakan salah satu pilihan dalam standar *point-to-point synchronous transmission*. Sistem CRC ini menghasilkan FCS 32-bit.

DAFTAR PUSTAKA

- Stallings, Williams, 1985. *Data and Computer communications*, Macmillan Publishing Company, New York.
- Ramabadrn, Tenkasi V. 1988, *A Tutorial on CRC computations*, IEEE Micro, August 1988, pp. 62 - 74
- Sait, Sadiq M dan Wasif hasan, 1995, *Hardware Design and VLSI Implementation of a Byte-wise CRC generator chip*, IEEE Transaction on Consumer Electronics, Vol. 41, No. 1, February 1995, pp. 195 - 200.